UNIVERSIDAD AUTONOMA DE MADRID

ESCUELA POLITECNICA SUPERIOR

Grado en Ingeniería de Tecnologías y Servicios de Telecomunicación

TRABAJO FIN DE GRADO

Detección de colisiones mediante procesado de vídeo

Sergio Avello Largo
Tutor: Álvaro García Martín
Ponente: José María Martínez

JUNIO 2021

Detección de colisiones mediante procesado de vídeo

AUTOR: Sergio Avello Largo
TUTOR: Álvaro García Martín
PONENTE: José María Martínez

Video Processing and Understanding Lab
Dpto. Tecnología Electrónica y de las Comunicaciones
Escuela Politécnica Superior
Universidad Autónoma de Madrid
Junio de 2021

Trabajo parcialmente financiado por el Gobierno de España bajo el proyecto TEC2017-88169-R (MobiNetVideo)

Resumen (castellano)

Este Trabajo Fin de Grado (TFG) consta de dos partes bastante diferenciadas. En la primera, se presenta una tecnología con tanto potencial como es la detección de accidentes mediante análisis de vídeo. Se comentan las posibles aplicaciones donde su uso sería de gran utilidad y el papel que podría jugar en un futuro cercano. A continuación, se introducen los conceptos básicos en los que se basa esta ciencia, como son el Aprendizaje Automático, la Visión Artificial, las redes neuronales artificiales o la detección de objetos. De esta forma, aunque el lector no sea un experto en la materia, será capaz de hacerse una idea del contexto en el que se engloba la detección de accidentes y podrá comprender el contenido de los siguientes capítulos.

La segunda parte es algo más técnica y nos centramos en un caso real de detección de accidentes que suceden alrededor de un vehículo. Basándonos en un modelo que ya había sido desarrollado y que utiliza el detector de objetos Faster R-CNN y una red neuronal VGG, realizamos un experimento modificando los módulos del detector de objetos (con EfficientDet) y de la extracción de *features* (con VGG16, AlexNet, DenseNet, ResNeXt). El algoritmo se entrena y testea con un *dataset* de *clips* en los que ocurren accidentes alrededor de un vehículo y es capaz de generar la probabilidad de accidente que existe para cada uno de los *frames*. A partir de esta probabilidad y tras un complicado procesamiento se obtienen los resultados de Area Under the Curve y el Tiempo al Accidente que permiten medir el rendimiento del experimento.

Por último, se comparan y analizan todos los resultados obtenidos para encontrar la mejor estructura del modelo dependiendo de la tarea a realizar.

Abstract (English)

This Bachelor Thesis consists of two quite different parts. In the first one, a technology with so much potential as accident detection by video analysis is presented. The possible applications where its use would be very useful and the role it could play in the near future are discussed. Then, the basic concepts on which this science is based are introduced, such as Machine Learning, Artificial Vision, artificial neural networks, or object detection. In this way, even if the reader is not an expert in the field, he/she will be able to get an idea of the context in which accident detection is included and will be able to understand the content of the following chapters.

The second part is somewhat more technical, and we focus on a real case of accident detection around a vehicle. Based on a model that had already been developed using the Faster R-CNN object detector and a VGG neural network, we conducted an experiment by modifying the object detector (with EfficientDet) and feature extraction modules (with VGG16, AlexNet, DenseNet, ResNeXt). The algorithm is trained and tested with a dataset of clips in which accidents occur around a vehicle and is able to generate the probability of accident that exists for each of the frames. From this probability and after a complicated processing, the results of Area Under the Curve and Tiempo al Accidente are obtained to measure the performance of the experiment.

Finally, all the results obtained are compared and analyzed to find the best model structure depending on the task to be conducted.

Palabras clave (castellano)

Detección de accidentes, Detector de objetos, Extracción de *features,* Probabilidad de accidente, Redes neuronales, Area Under the Curve, Tiempo al Accidente.

Keywords (inglés)

Accident Detection, Object Detector, Feature Extraction, Accident Probability, Neural Networks, Area Under the Curve, Time to Accident.

Agradecimientos

Quiero darle las gracias a mi tutor Álvaro por su paciencia y por toda la ayuda que me ha ofrecido a lo largo del desarrollo del trabajo.

Agradecer también a mi familia, ellos me han apoyado en todo momento durante los cinco años de carrera y a todos mis amigos en especial Rebeca, Manu y Rodri. Sin ellos quizás estaría en el mismo sitio, pero no me sentiría igual.

Gracias a todos.

INDICE DE CONTENIDOS

INDICE DE FIGURAS

INDICE DE TABLAS

1 Introducción

1.1 Motivación

Esta memoria de trabajo de fin de grado es el broche final al duro esfuerzo realizado conjuntamente por mi tutor y por mí durante dos años con la intención de finalizar satisfactoriamente mis estudios de grado. A principios del curso 2019/20, me puse en contacto con Álvaro García para que fuera mi tutor debido a lo estimulante que me había resultado su asignatura Tecnologías de Vídeo del tercer año de carrera. Al contar con su colaboración, buscaba que mi trabajo de fin de grado relacionara algunos de los conceptos más interesantes que he descubierto a lo largo de mi trayectoria universitaria como son la Inteligencia Artificial, el Aprendizaje Profundo y el procesado de vídeo. Después de varias conversaciones, llegamos a la conclusión de que sería interesante combinar estas ideas y aplicarlas a un tema tan actual como es la detección de colisiones mediante procesado de vídeo para anticipar posibles accidentes alrededor de un vehículo.

En los últimos años, la tecnología enfocada a la seguridad vial se ha centrado en los asistentes a la conducción como son el detector de ángulo muerto, la frenada de emergencia o el asistente de aparcamiento. Entre 2015 y 2019, la prestigiosa organización de consumidores de EEUU Consumer Reports encuestó a sus miembros y evaluó los datos de más de 72.000 vehículos. El 57% de los encuestados declaró que al menos un sistema de asistencia les ha evitado un accidente [1].

El siguiente gran avance en el mundo del automóvil debe ser la conducción autónoma, que se basa en los asistentes a la conducción que han sido mencionados anteriormente y a una extensa cantidad de técnicas complejas para replicar las capacidades humanas de manejo, control o decisión. La conducción autónoma, tal y como se detalla en la Figura 1, se organiza en 6 niveles, desde el nivel 0 en el que todas las acciones las realiza el conductor, hasta el 5 que supone la automatización completa en el que no existe la figura del conductor. En la actualidad, en los concesionarios ya es posible encontrar vehículos de nivel 2 y 3, donde existe una simbiosis entre conductor y sistemas de automatización [2].

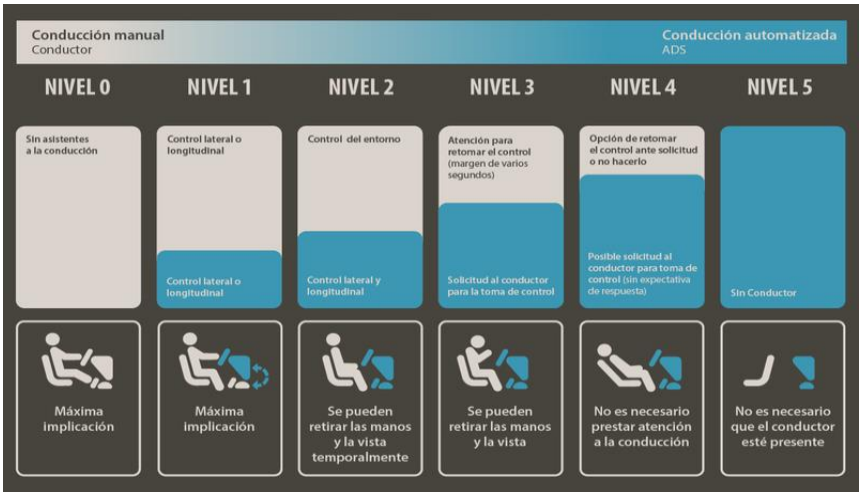

Figura 1: Niveles de conducción autónoma. Procedente de [3]

Para contextualizar el impacto que podría generar una conducción autónoma óptima y fiable, merece la pena destacar los siguientes datos: Según la Dirección General de Tráfico, solo en España, en 2019 (no se mencionan los datos de 2020 ya que debido a la pandemia los mismos no son del todo realistas), se notificaron 141.104 accidentes de tráfico con víctimas, los cuales ocasionaron 1.755 víctimas mortales, como se puede observar en la Figura 2. 519 de estas muertes se produjeron en las vías urbanas (esta información se destaca ya que, como más tarde se verá, nuestro experimento ha sido realizado en este tipo de vías) [4].

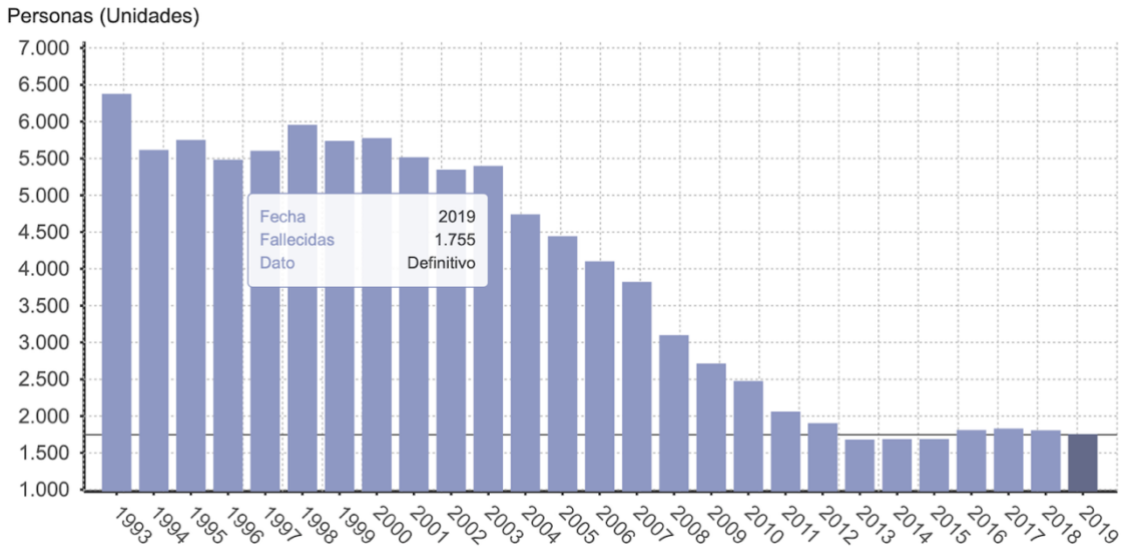

Figura 2: Muertes por accidentes de tráfico al año en España. Procedente de [5]

Estas cifras son impactantes año tras año y más si las extrapolásemos a todo el mundo, aunque muchas veces acaban pasando desapercibidas. Sin embargo, nos permite hacernos una idea de lo que podríamos mejorar en términos de seguridad vial si se implementara una tecnología fiable que ayude al conductor o incluso lo sustituya a la hora de conducir un vehículo.

Una vez presentado este contexto queda claro que existe una estrecha relación entre la tecnología y el mundo de la conducción autónoma. Sin embargo, esta combinación tiene multitud de variantes y posibilidades. Por ello, en este trabajo de fin de grado nos centraremos en un concepto más específico, pero cuya importancia en los ámbitos comentados durante este capítulo es y será vital. Se trata de la detección y la anticipación de colisión mediante procesado de vídeo, las cuales comienzan ya a estar presentes en el día a día del gran público y cuyo potencial para poder ser utilizado en diversas aplicaciones es enorme.

1.2 Objetivos

Como se ha comentado en el apartado anterior, uno de los grandes retos de la Inteligencia Artificial (IA) durante los próximos años es conseguir que un coche sea "conducido" por un agente inteligente no humano. Desde inicios de siglo se han logrado numerosos avances significativos, sin embargo, algunos aspectos cruciales como el proceso de aprendizaje de la entidad inteligente o la conducción segura en situaciones

complicadas con otros conductores humanos aún tienen mucho margen de mejora. Los humanos vamos desarrollando nuestras habilidades en la conducción a medida que nuestro tiempo al volante de un vehículo aumenta, después de este periodo de aprendizaje somos capaces de reconocer situaciones potencialmente peligrosas y sabemos cómo reaccionar ante ellas. En cambio, para una IA aprender a identificar y ser capaz de anticipar con cierto margen un accidente se torna una tarea ciertamente complicada [6].

Los accidentes en los que están involucrados los vehículos suelen suceder de forma muy repentina y su naturaleza puede ser muy diversa. Existen números estudios cuyo objetivo es predecir colisiones en los que interviene el vehículo desde el que se está realizando el experimento. Sin embargo, una situación potencialmente peligrosa no sólo depende de un vehículo y su conductor, sino que también juegan un papel fundamental el entorno y el resto de los vehículos que lo rodean. Es por ello, que este TFG tiene como objetivo profundizar en un concepto algo más novedoso como es la detección de accidentes mediante procesado de vídeo que ocurren en el entorno de un vehículo, que permitirá anticipar estas situaciones y evitar que afecten al propio vehículo. Conoceremos cuáles son los algoritmos o las redes más utilizadas actualmente para esta tarea y compararemos su comportamiento y resultado.

1.3 Organización de la memoria

La memoria consta de los siguientes capítulos:

- **Capítulo 1:** Motivación, objetivos y organización de la memoria.
- **Capítulo 2:** Estado del arte.
- **Capítulo 3:** Diseño.
- **Capítulo 4:** Desarrollo.
- **Capítulo 5:** Integración, pruebas y resultados.
- **Capítulo 6:** Conclusiones y trabajo futuro.
- **Bibliografía**
- **Glosario**

2 Estado del arte

2.1 Introducción

La detección de accidentes mediante procesado de vídeo es un concepto realmente complejo, con multitud de enfoques y que se encuentra es su máximo apogeo, tanto por su utilidad como por su potencial. Por ende, antes de profundizar en los temas más técnicos de este trabajo, merece la pena presentar las ideas más generales de las que se nutre la detección de accidentes mediante procesado de vídeo para tener una percepción global de lo que vamos a tratar.

2.1.1 Inteligencia Artificial

En 1950, Alan Turing presenta en un ensayo el test de Turing y se convierte en el primer científico que hace referencia a la Inteligencia Artificial. Existen numerosas versiones de este experimento, una de las primeras consistía en colocar a una máquina y a un humano a contestar con respuestas escritas (así el habla no influye) las preguntas de un interrogador durante unos cinco minutos. Una vez transcurrido ese tiempo, en caso de que el interrogador no fuese capaz de distinguir quién era la máquina y quién el humano, se podía considerar que la máquina era "inteligente" [7].

Por otra parte, es interesante conocer las diferencias entre una IA y un programa/aplicación informático/a ya que a veces la frontera que los separa es un poco difusa. Un programa informático es simple y llanamente una consecución de instrucciones que le indican a una máquina como debe actuar. Estas instrucciones cubren todos los posibles escenarios que puedan suceder, incluso los errores, por lo que la máquina no tiene que decidir, únicamente debe seguir las "órdenes". Sin embargo, una IA sólo recibe una información de entrada y con esos datos, y sin recibir ningún tipo de instrucción extra debe ser capaz por si misma de obtener los resultados [8].

Para conseguir realizar una tarea con éxito, una IA debe replicar el comportamiento del pensamiento humano, por ello se trata de una tecnología realmente revolucionaria. Primero aprende una tarea e intenta ponerla en práctica, obviamente al principio comete muchos errores y habrá que indicarle lo que está haciendo mal, pero a medida que su entrenamiento aumente, también lo harán sus aciertos, hasta finalmente poder llevar a cabo una tarea sin ningún tipo de ayuda. Este método de trabajo (aprendizaje, entrenamiento y obtención de resultados) es la característica principal de las IA y se aplica en muchos de los algoritmos que veremos en los próximos capítulos [9].

2.1.2 Aprendizaje Automático (*Machine Learning*)

El Aprendizaje Automático es un parte fundamental de la Inteligencia Artificial y por ello entronca a la perfección con lo explicado en el apartado anterior, ya que se define como la capacidad que tiene una máquina o un software para aprender por sí misma [10].

Una definición quizás más técnica pero muy esclarecedora fue enunciada por el científico informático e investigador Tom M. Mitchell en su obra Machine Learning (1997):

"Un programa informático aprende de la experiencia E con respecto a alguna clase de tarea T y medida de rendimiento P, si su rendimiento en la tarea T medido por P mejora con la experiencia E [11]"

Un programa informático que aprende presenta unos parámetros que son modificables y que permiten variar su comportamiento. Un algoritmo de aprendizaje es el encargado de ajustar estos parámetros a medida que va recibiendo nueva información, consiguiendo así que el programa vaya aprendiendo y optimizando su rendimiento [12].

Existen dos tipos de Aprendizaje Automático:

- **Supervisado:** En este caso los datos de entrada con los que el algoritmo se entrena están etiquetados (se conoce el resultado correcto para cada dato de entrada), con ello el algoritmo aprende y genera un modelo que es capaz de predecir correctamente el valor de salida para un dato sin etiquetar. Este tipo de aprendizaje suele aplicarse en problemas de clasificación (predecir el valor de una variable discreta) o de regresión (predecir el valor de una variable continua) [13].

- **No supervisado:** Los datos de entrada dejan de estar etiquetados por lo que el algoritmo procura relacionar de alguna forma la información de entrada y conseguir así simplificar el análisis. El aprendizaje no supervisado se utiliza principalmente en problemas de *clustering* o agrupamiento [13].

2.1.3 Aprendizaje Profundo (*Deep Learning*)

El Aprendizaje Profundo es una de las ramas del frondoso árbol que representa el Aprendizaje Automático. En ambos casos los algoritmos utilizan enormes cantidades de datos para aprender y mejorar su rendimiento. Sin embargo, las técnicas que emplean para aprender son muy diferentes. En el caso del Aprendizaje Profundo basa su aprendizaje en el procesamiento de los datos mediante redes neuronales artificiales. Este nuevo concepto que aparece aquí por primera vez en todo el documento es uno de los grandes pilares en los que se sustenta este trabajo y será tratado en los siguientes capítulos [14][15].

Mediante el uso de estas redes es posible generar unas arquitecturas de aprendizaje por capas (de ahí el término profundo) con infinitas posibilidades de configuración según la tarea por realizar y que permiten tanto procesar como manejar grandes volúmenes de datos de forma más rápida.

Es debido a estas características por lo que el Aprendizaje Profundo supera normalmente al Aprendizaje Automático en cuanto a rendimiento y ha conseguido que hayan dejado de ser utopías algunas tareas que hasta algunos años atrás parecían imposibles como son el reconocimiento de voz, la detección de objetos en imágenes o la interpretación de textos [14].

2.1.4 Visión Artificial (*Computer Vision*)

La Visión Artificial o Visión por Ordenador es otro de los campos que engloba el Aprendizaje Automático. Como su nombre indica, esta ciencia estudia y trabaja con máquinas que son capaces de "ver", donde "ver" significa tener la capacidad de entender el contenido y extraer la información deseada de una imagen digital para resolver una determinada tarea [16].

Esta tecnología vanguardista se encuentra enormemente relacionada con el Aprendizaje Automático y el Aprendizaje Profundo y se apoya en ellos para crear soluciones especializadas en el ámbito del procesado de imágenes digitales.

Actualmente, la gran mayoría de dispositivos electrónicos disponibles ya son capaces de generar contenido digital y existe una gran demanda a la hora de procesar toda esta información para cualquier tipo de objetivo. Debido a esta necesidad y a la innumerable cantidad de aplicaciones que puede tener esta ciencia, muchos sectores e industrias están invirtiendo en Visión Artificial [17][18].

2.2 Redes neuronales artificiales

Comenzamos ahora con uno de los conceptos más importantes del trabajo. En los próximos capítulos el tema será descrito en detalle y se explicarán las diferentes redes que hemos utilizado en nuestro experimento junto con el papel que desempeñan en el mismo.

2.2.1 Definición

Las redes neuronales artificiales son una de las técnicas más utilizadas en el Aprendizaje Profundo y procuran imitar el funcionamiento del sistema nervioso de los seres vivos para conseguir aprender y ser capaces de predecir un determinado resultado. Al igual que el sistema nervioso está formado por conexiones de neuronas que trabajan conjuntamente, una red neuronal artificial se basa en un conjunto interconectado de neuronas artificiales [19][20][21][22].

2.2.2 Perceptrón

La neurona artificial, también llamada Perceptrón, es el elemento básico a partir del cuál nacerían y se potenciarían las redes neuronales artificiales. Un Perceptrón es una entidad que calcula una suma ponderada de sus datos de entrada y a continuación le aplica una función de activación para obtener así un resultado que se convierte en la salida de la neurona [23][10]. Su arquitectura se presenta en la Figura 3.

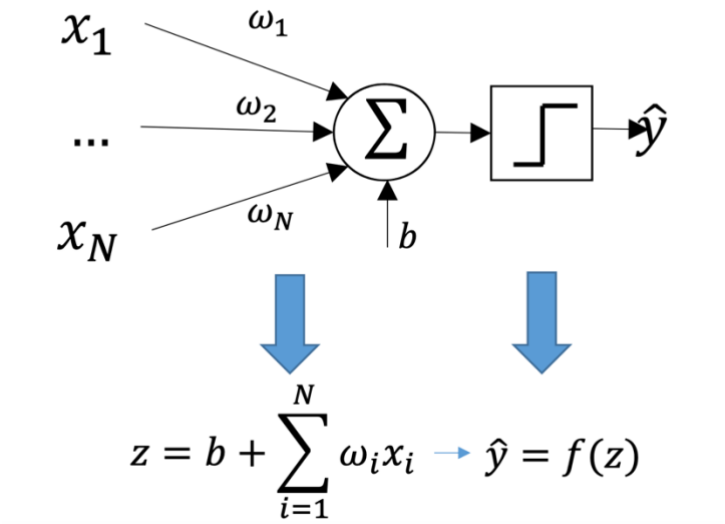

Figura 3: Perceptrón. Procedente de [24]

El siguiente paso al Perceptrón es el Perceptrón Multicapa. Este tipo de red, como se observa en la Figura 4, tiene al menos una capa de nodos entre las capas de entrada y salida, lo que permite aumentar la potencia del algoritmo. Al añadir más capas la complejidad de la red crece, sin embargo, es capaz de procesar conjuntos de datos mucho más complejos y resolver tareas que serían imposibles de tratar con un Perceptrón simple [25].

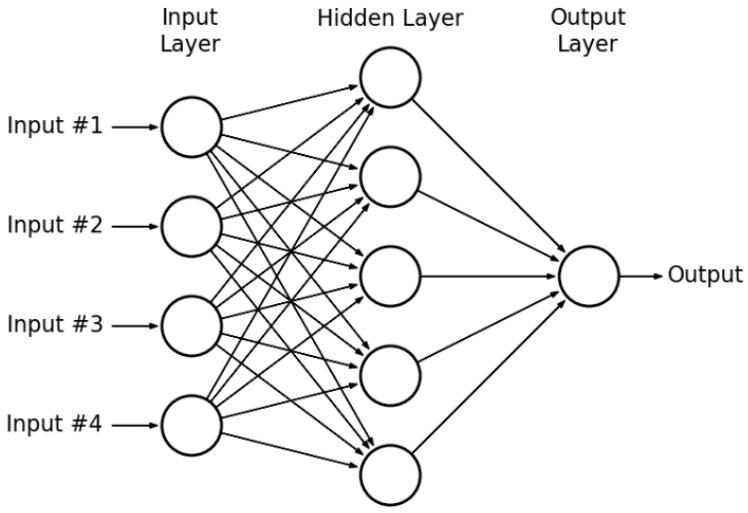

Figura 4: Perceptrón Multicapa. Procedente de [26]

Los conceptos más importantes de la explicación anterior y que son comunes a todas las redes posteriores al Perceptrón son los siguientes:

- **Capas:** El número de capas de una red indica la complejidad de la misma. Con cada capa, la red modifica los datos de entrada y crea una nueva representación. Existen tres tipos de capas [27]:

- **Capa de entrada:** Es la primera capa de la red, se encarga introducir los datos de entrada en la red. En ella no se realiza ningún tipo de procesamiento.

- **Capas ocultas:** Puede haber tantas como el diseñador desee y sus características son muy variables. En estas capas es donde se realiza todo el procesamiento de la red.

- **Capa de salida:** Se encarga de devolver el resultado después de todo el procesamiento de la red.

- **Pesos:** Son las *w* de la Figura 3. Permiten ponderar los datos de entrada y por lo tanto representan la importancia que tiene la entrada en relación con la salida. En un principio, estos pesos eran asignados manualmente por un científico, aunque como veremos a lo largo de este capítulo, la mejora de los equipos tecnológicos y la aparición de algunos algoritmos ha permitido que las redes puedan "aprender" solas al actualizar estos pesos de forma autónoma [25].

- **Función de activación:** Es la *f(z)* de la Figura 3. Esta operación se aplica, en general, al final del proceso de cada neurona. Existe una gran variedad y se eligen en función del objetivo de la red [28][29]. Las más utilizadas se muestran en la Figura 5 y se detallan a continuación:

    - **Sigmoide:** Devuelve un valor entre 0 y 1. Suele utilizarse en problemas de predicción o clasificación binaria.

    - **Tangente Hiperbólica:** Similar a la sigmoide, pero entre valores -1 y 1. Suele utilizarse en redes multicapa.

    - **ReLu (Rectifier Linear Unit):** Elimina los valores negativos. Suele utilizarse en tareas de Visión Artificial o Reconocimiento de Voz. Existen varios tipos de funciones similares como Leaky ReLu o Exponential Lu.

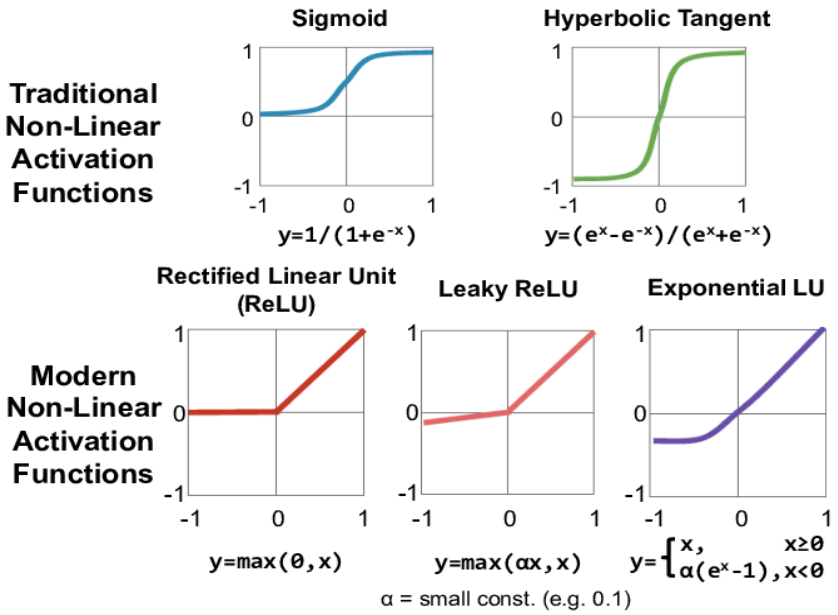

Figura 5: Diferentes funciones de activación. Procedente de [29]

2.2.3 Retropropagación (*Backpropagation*)

El proceso aprendizaje de una red neuronal artificial consiste en realizar pequeñas modificaciones en los pesos de las entradas de todas las neuronas hasta llegar a la combinación exacta que devuelva el resultado esperado. Durante los primeros años de la segunda mitad del siglo XX, este proceso era bastante tedioso y unido con la poca capacidad de procesamiento de las máquinas de la época, provocó que las redes neuronales artificiales cayeran un poco en el ostracismo [30][19].

Sin embargo, en 1986, David Rumelhart, Geoffrey Hinton y Ronald Williams presentaron en un artículo [31] el algoritmo de Retropropagación (*Backpropagation*). Este algoritmo cambió por completo el enfoque que se tenía de las redes neuronales artificiales y se ha convertido en la piedra angular del Aprendizaje Automático moderno.

La popularidad de este algoritmo tiene mucho sentido, ya que, a pesar de ser una tecnología bastante sencilla, su rendimiento es francamente potente. El concepto general es bastante simple: definir una función de pérdidas (que compara la salida con el resultado esperado) y aplicar la técnica de descenso por gradiente hasta encontrar un conjunto de pesos que genere el resultado óptimo para una determinada tarea [30].

2.2.4 Redes Neuronales Convolucionales

Las redes e ideas que han sido explicadas en los apartados anteriores pueden ser utilizadas en una enorme variedad de tareas, sin embargo, nuestro trabajo está enfocado hacia un concepto muy especial: el procesado de vídeo. La principal particularidad que se puede observar en comparación con otras tareas es que en vez trabajar con variables tanto discretas como continuas se trabaja con imágenes que, al fin y al cabo, son matrices de un determinado tamaño y con una determinada combinación de valores (pixeles).

La mejor opción para manejar y procesar conjuntos de datos de gran dimensionalidad (como son las imágenes y los vídeos) son las redes neuronales convolucionales. Este tipo de tecnología procura imitar el funcionamiento del córtex visual del cerebro para dotar a una máquina con la capacidad de "ver". Por esta razón es muy utilizada en tareas de Visión Artificial como la detección de objetos o la segmentación de imágenes y, por su puesto, tendrá un papel fundamental en nuestro estudio [31].

Seguramente, los componentes principales de las redes neuronales convolucionales sean los filtros o *kernels*. Estos filtros son matrices cuadradas de reducida dimensión (normalmente 3x3 o 5x5), los cuales al aplicarlos a lo largo y ancho de la matriz de entrada y dependiendo de su diseño, son capaces de detectar diferentes características o patrones que estén presentes en la imagen original como bordes o cambios de luminosidad [32].

La arquitectura de las redes neuronales convolucionales se basa en el apilamiento de diferentes tipos de capas y operaciones dependiendo del objetivo que se busque y vamos a proceder a explicarla [31][32][33][34][35][36]:

- **Capa de entrada:** La red toma como entrada una matriz que representa los pixeles de una imagen. Si la imagen es en escala de grises, sólo entrará una matriz, sin embargo, si es en color, la entrada serán tres matrices que representan los canales R, G y B. Para que nos hagamos una idea del tamaño de las redes neuronales

convolucionales y su capacidad de procesamiento, merece la pena mencionar que si tomamos una imagen de 16 x16 pixeles (escala de grises), la capa de entrada tendrá 16 x 16 = 256 neuronas.

- **Preprocesamiento:** Antes de seguir con el procesamiento en la red puede ser interesante normalizar los valores de la matriz original para simplificar las operaciones. El valor de los píxeles está entre 0 y 255, por lo que se divide entre 255 para conseguir pixeles entre 0 y 1.

- **Capas convolucionales:** Es aquí donde ocurre el proceso más característico de este tipo de redes: la convolución. Esta operación, no es más que aplicar sobre la matriz (o matrices si es RGB) de entrada de la capa, los filtros explicados en la página anterior para obtener nuevas matrices (mapas de características) que representan los patrones detectados.

  La operación de convolución, matemáticamente hablando, consiste en multiplicar escalarmente el *kernel* por una región del mismo tamaño de la matriz de entrada, obteniendo así el valor de la matriz de salida en esa posición. Este proceso se repite desplazando el *kernel* de izquierda a derecha y de arriba a abajo hasta rellenar todas las posiciones de la matriz de salida. Para facilitar la tarea de convolución, la región seleccionada de la matriz de entrada suele rodearse con ceros, esta técnica se conoce como Zero Padding. En el caso de que sean tres canales, el resultado sería el sumatorio del resultado para cada una de las matrices (R, G y B). Para que esta explicación sea más clara se presenta un ejemplo en la Figura 6:

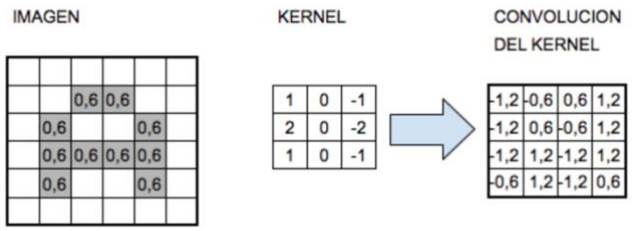

Figura 6: Ejemplo de filtrado. Procedente de [35]

El valor de los cuatro primeros pixeles (fila superior) de la matriz resultado de la Figura 6 proviene de:

$$1 \times 0 + 0 \times 0 + -1 \times 0 + 2 \times 0 + 0 \times 0 + -2 \times 0.6 + 1 \times 0 + 0 \times 0.6 - 1 \times 0 = -1.2$$

$$1 \times 0 + 0 \times 0 + -1 \times 0 + 2 \times 0 + 0 \times 0.6 + -2 \times 0.6 + 1 \times 0.6 + 0 \times 0 - 1 \times 0 = -0.6$$

$$1 \times 0 + 0 \times 0 + -1 \times 0 + 2 \times 0.6 + 0 \times 0.6 + -2 \times 0 + 1 \times 0 + 0 \times 0 + -1 \times 0.6 = 0.6$$

$$1 \times 0 + 0 \times 0 + -1 \times 0 + 2 \times 0.6 + 0 \times 0 + -2 \times 0 + 1 \times 0 + 0 \times 0.6 + -1 \times 0 = 1.2$$

Normalmente, en una capa convolucional se utilizan más de un filtro a la vez, un ejemplo que se utiliza habitualmente es 32 y por lo tanto se obtienen 32 mapas de características. A continuación, suele aplicarse una función de activación (la más popular es ReLu: $f(x) = \max(0, x)$) para que el resultado sea más esclarecedor.

- **Capa de pooling:** Una vez llegados a este punto, si nos enfrentáramos a un caso normal, por ejemplo, procesar una imagen RGB de 256 x 256 con 32 filtros de 3 x 3 x 3, estaríamos utilizando en esta capa más de dos millones de neuronas.

  Como se puede entender, esta cantidad de neuronas comienza a ser muy grande y si se siguiera aplicando nuevas capas convolucionales, el número se dispararía. Por ello, existe una técnica conocida como submuestreo que permite reducir la cifra de neuronas de la siguiente capa sin perder las características principales que se han detectado anteriormente. Por ejemplo, tomando una sección de 2 x 2 pixeles y aplicando submuestreo es posible rebajar la cantidad de neuronas a un cuarto de la cifra original sin, prácticamente, perder la información relevante. Los casos más conocidos son Max Pooling (se selecciona el máximo) y Average Pooling (se calcula la media). Se presenta un ejemplo de ambos casos en la Figura 7.

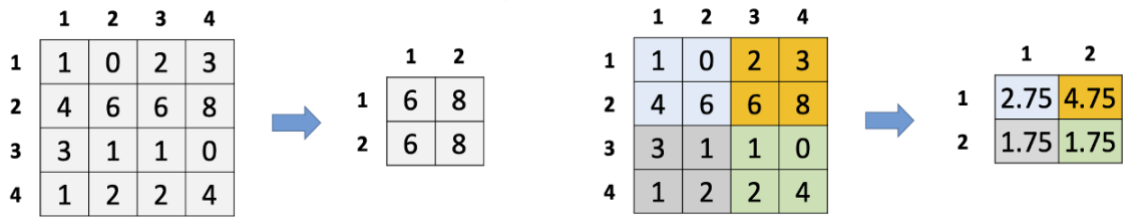

Figura 7: Ejemplos de Max-Pooling (izq.) y Average-Pooling (dcha.). Procedente de [37]

Lo explicado hasta ahora se podría considerar como el núcleo principal de las redes neuronales convolucionales y queda muy bien reflejado en la Figura 8:

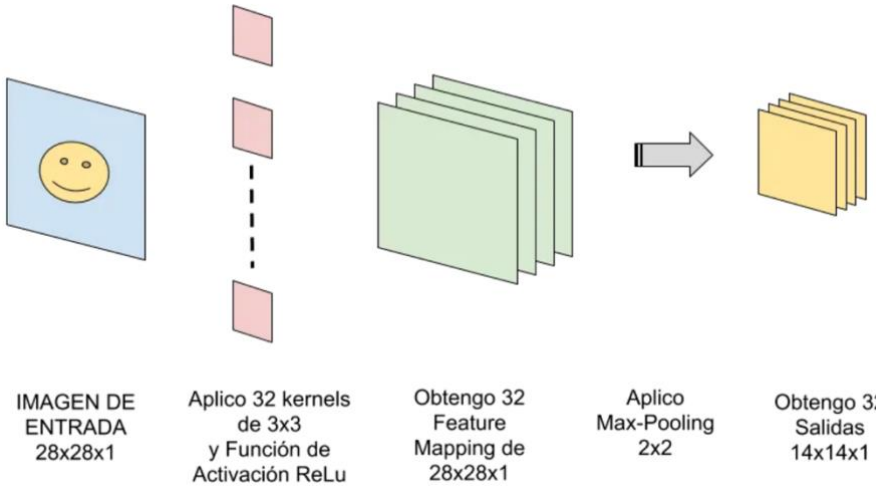

Figura 8: 1ª etapa de una red neuronal convolucional. Procedente de [35]

Otra de las características principales de las redes neuronales convolucionales es que las capas que la forman siguen una estructura jerárquica. Esto quiere decir que las primeras capas convolucionales se centran es detectar patrones simples, como líneas, bordes o esquinas, y a medida que la profundidad aumenta se van especializando hasta reconocer formas más complejas, como objetos o personas. Es por ello por lo que una vez terminada esta primera etapa de convolución lo habitual es seguir añadiendo capas de

procesamiento hasta reducir las dimensiones de las matrices de salida al máximo (hasta el tamaño del filtro).

La estructura de las redes neuronales convolucionales continúa conectando el final de la etapa de capas convolucionales con una red neuronal "tradicional" con el objetivo de clasificar los resultados. Para ello, el último conjunto de matrices resultantes después del procesamiento de capas convolucionales se "aplana" para convertirse en una capa de una red "normal". El último paso consiste en aplicar una función denominada Softmax que desemboca en la capa final que contiene tantas neuronas como clases queramos clasificar. Esta función Softmax devuelve la probabilidad de acierto para cada clase. Para terminar, la Figura 9 ofrece una visión global de lo que es una red neuronal convolucional:

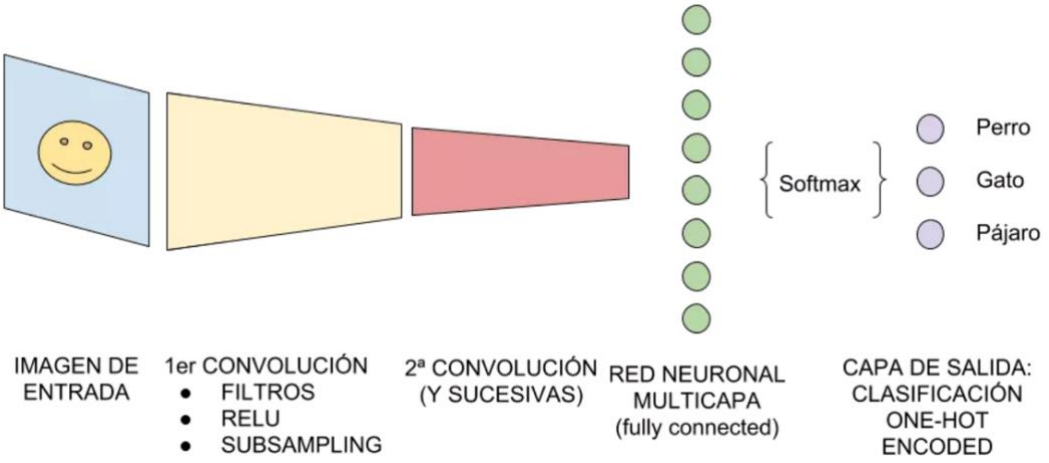

Figura 9: Arquitectura general de una red neuronal convolucional. Procedente de [35]

2.3 Detección de objetos

Para finalizar con el capítulo de estado del arte, se presenta una de las tareas más avanzadas en cuanto al procesamiento de imágenes mediante redes neuronales: la detección de objetos. Este concepto juega un papel fundamental en la Visión Artificial y por extensión en nuestro trabajo [38].

La detección de objetos es una tarea ciertamente desafiante y difícil puesto que depende de factores muy complejos, como la luminosidad de una escena, el tamaño de los objetos, las oclusiones… Sin embargo, esta tecnología, se aplica ya a multitud de áreas, véanse la conducción autónoma, la robótica, la video vigilancia… [33].

El objetivo final de la detección de objetos es ser capaz de localizar la posición de los diferentes objetos que aparezcan (coches, animales, personas…) y recuadrarlos al procesar una imagen. El resultado que se suele obtener son las coordenadas (en pixeles) donde se encuentran estos recuadros. Algunos de los algoritmos más populares que suelen utilizarse son Faster R-CNN [39] o YOLO9000 [40][41].

3 Diseño

3.1 Introducción

En esta nueva sección procederemos a explicar cómo está enfocado nuestro experimento, los modelos en los que se basa y cómo se aplican los conceptos explicados anteriormente.

El desarrollo de los algoritmos para la detección de accidentes que ocurren en el entorno de un vehículo mediante procesado de vídeo ha crecido exponencialmente en los últimos años y la oferta existente actualmente es muy extensa. Por consiguiente, cuando comenzó a gestarse este proyecto, nuestra idea fue utilizar un algoritmo reciente que ya estuviera probado en la tarea de detección y cuyo rendimiento fuera razonable, para luego modificar ciertos de sus módulos y realizar una comparación entre los resultados obtenidos.

Tras varios meses de búsqueda nos decidimos por el modelo propuesto por Fu-Hsiang Chan, Yu-Ting Che, Yu Xiang y Min Sun en el *paper* "*Anticipating Accidents in Dashcam Videos"* [6], ya que cumplía con todos nuestros requisitos y además era el que mejor se adaptaba a los medios de los que disponíamos. A continuación, se explicará de forma general el funcionamiento de este modelo.

3.2 Dynamic-Spatial-Attention Recurrent Neuronal Network

En el *paper* que acaba de ser citado, se detalla minuciosamente cada aspecto del experimento que realizaron sus autores. Sin embargo, nosotros nos centraremos en diseccionar los aspectos técnicos del algoritmo original para que luego queden claros los cambios que se realizaran en nuestro experimento y que serán explicados en el Capítulo 4.

El núcleo del algoritmo que desarrollaron lo denominaron Dynamic-Spatial-Attention Recurrent Neuronal Network (DSA-RNN), que en definitiva se trata de una red neuronal recurrente que aprende a distribuir dinámicamente la atención (posibilidad de accidente) sobre los diferentes objetos detectados en vídeos (grabados desde el salpicadero de un vehículo), para encontrar evidencias sutiles de un posible accidente que ocurra alrededor del vehículo y ser capaz así de predecirlos de la manera más robusta posible.

Para alimentar de forma correcta la red es necesario extraer la información necesaria de los vídeos que estamos utilizando como *dataset*. Para ello, se utilizan dos tecnologías principalmente.

- **Faster R-CNN:** Se trata de un detector de objetos vanguardista presentado en 2015. Su misión es delimitar los objetos (vehículos, personas, señales…) presentes en un *frame* mediante un rectángulo y devolver las coordenadas de las esquinas de estas "cajas" para que puedan ser utilizadas por la siguiente etapa [42]. El detector está configurado para que como máximo detecte 19 objetos.

- **VGG:** Esta red neuronal convolucional pre-entrenada es una de las más populares de los últimos años y existen varias versiones de ella. En este experimento se utiliza

para procesar cada una de las detecciones de un *frame* provenientes del detector Faster-RCNN además del *frame* completo y obtener por cada detección un vector de 4.096 *features* que, a posteriori, sirve como entrada para la DSA-RNN [43].

El algoritmo propuesto recibe las 4.096 *features* de cada objeto detectado y de la imagen completa que se obtienen como salida de la red VGG, las procesa y combina para reunir la máxima información posible y por último devuelve la probabilidad de accidente alrededor del vehículo de cada *frame*. Si esta probabilidad es mayor que un umbral, el sistema notifica que se producirá un accidente alrededor del vehículo "x" segundos antes de que suceda realmente, siendo x la variable Time to Accident que se definirá más adelante. La arquitectura del algoritmo puede observarse en la Figura 10.

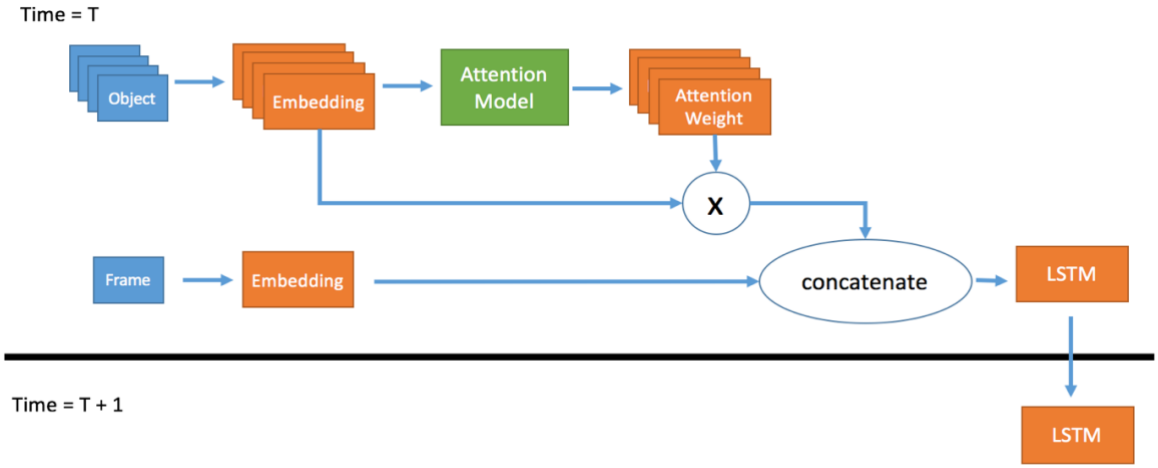

Figura 10: Diagrama de flujo del modelo. Procedente de [44]

El procesamiento de las *features* se realiza en los siguientes módulos:

- **Dynamic Spatial Attention (DSA):** Se basa en el concepto propuesto en el *paper* "*Show, attend and tell: Neural image caption generation with visual attention*" [45] para centrar la atención de la RNN únicamente en los objetos que encontró el detector Faster R-CNN, otorgándole a cada detección un "peso" dependiendo de su relevancia respecto a la del instante anterior. La principal diferencia que existe con el mencionado *paper* es que en vez de procesar la información correspondiente a un solo *frame*, los cálculos se realizan teniendo en cuenta una secuencia temporal de varios *frames*. Es aquí donde juega un papel fundamental la red neuronal recurrente, como veremos a continuación [6].

- **Recurrent Neural Network (RNN):** Esta clase de redes neuronales trabaja con secuencias de tamaño variable en lugar de con entradas de un tamaño fijo como estábamos acostumbrados. Su principal característica es que se trata de un sistema dinámico, es decir, que el procesamiento de una entrada en un determinado instante depende de la operación realizada en el instante anterior y en ocasiones de la que se realizará en el instante futuro, lo cual permite al modelo mantener información de las entradas de instantes pasados y poder así encontrar correlaciones que existan entre entradas de diferentes momentos temporales. Podría decirse que las redes neuronales recurrentes tienen la capacidad de "predecir el futuro" y es por ello que su rendimiento es excelente en tareas como la conducción autónoma [46][47]. La arquitectura de una RNN se presenta en la Figura 11.

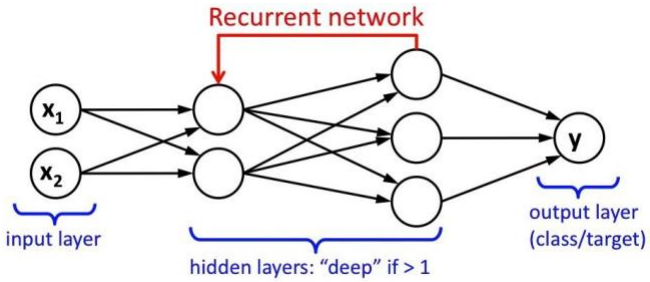

Figura 11: Arquitectura de una RNN. Procedente de [48]

Normalmente el principal problema de las redes neuronales recurrentes aparece cuando se enfrentan a tareas cuyo resultado depende de las entradas de numerosos instantes atrás, y este es precisamente el caso de la detección de accidentes. Para solucionarlo, se propuso sustituir la función de activación de cada neurona por una célula *Long Short-Term Memory* (LSTM) [49].

- **Long Short-Term Memory (LSTM):** El objetivo de este tipo de células es mantener la información a lo largo del tiempo. La LSTM, mediante unas estructuras llamadas "puertas", es capaz de modificar la información que posee. Las "puertas" son simplemente operaciones (sigmoides, multiplicaciones…) que permiten añadir o eliminar información de la célula [50].

3.2.1 Ejemplo visual del funcionamiento del algoritmo

Para completar la explicación de todo este capítulo, en esta sección presentamos un ejemplo visual del procesamiento que realiza el algoritmo para cada vídeo. Tal y como ya hemos comentado, por cada *frame* del vídeo, el detector recuadra todos los objetos que encuentre en la escena, mediante la red de extracción de *features* se obtiene un vector de tamaño 4.096 por cada detección y, finalmente, con esta información, se realizan diferentes operaciones para obtener la probabilidad de accidente alrededor del vehículo que existe en cada *frame*. En la gráfica de la Figura 12 se puede observar cómo evoluciona la probabilidad a lo largo del *clip* 489 del *dataset*.

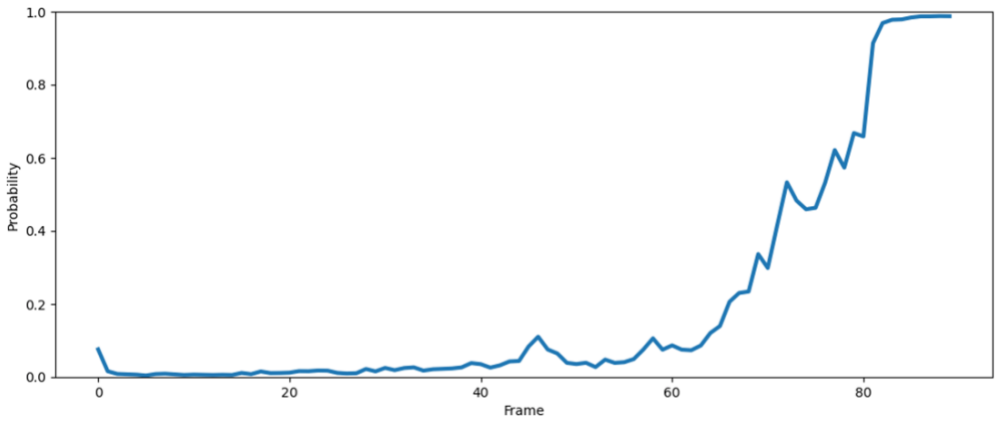

Figura 12: Probabilidad de accidente en torno al vehículo por cada *frame* en el *clip* 489. Hasta el *frame* 60 la probabilidad de accidente es mínima, del *frame* 60 al 80 la probabilidad crece prácticamente hasta 1 y a partir de 80, el accidente se confirma. En la Figura 14 puede verse lo que ocurre en imágenes.

En la Figura 13 mostramos las capturas de los *frames* 31, 67 y 83 al procesar el *clip* 489. En cada captura aparecen diferentes rectángulos que seleccionan cada detección y un número decimal entre 0 y 1 que corresponde a su probabilidad de accidente. Los rectángulos rojos indican que para ese instante el algoritmo no prevé ningún problema y si es verde significa que la probabilidad de accidente de la escena supera el umbral (en este caso 0.4) y por lo tanto existe una situación potencialmente peligrosa alrededor del vehículo. En la Figura 12 queda claro que hasta el *frame* 60 no existe ningún peligro, sin embargo, a partir de ese momento, la probabilidad comienza a crecer y alrededor del *frame* 80 el accidente es inminente (lo cual se corrobora en la captura del *frame* 83 de la Figura 13).

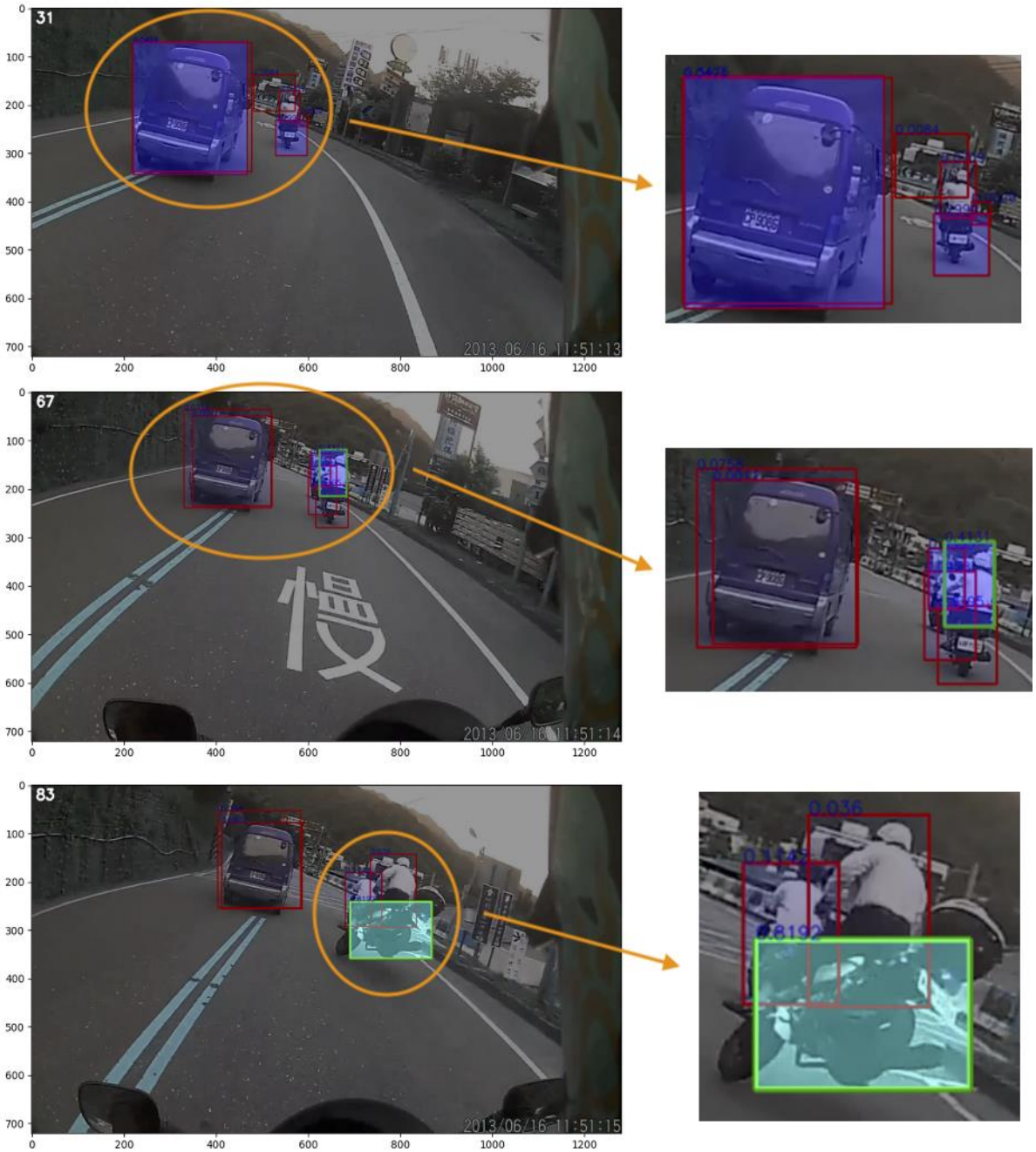

Figura 13: *Frames* 31, 67, 83 del procesamiento del *clip* 489, donde se observa la relación que existe entre lo que ocurre en las imágenes y la evolución de la probabilidad de accidente por cada detección.

4 Desarrollo

4.1 Introducción

Tal y como se mencionó al inicio del capítulo anterior, el planteamiento de nuestro experimento era basarnos en el modelo de la DSA-RNN propuesto en el *paper* "*Anticipating Accidents in Dashcam Videos"* [6] para luego modificar ciertos módulos de su arquitectura y comparar los resultados obtenidos en cada caso. A lo largo de los próximos apartados se explicarán los cambios realizados y como afectan a la estructura del algoritmo.

4.2 Detector de objetos

Todos los detectores de objetos tienen en general el mismo objetivo: encontrar y seleccionar con la mayor precisión posible todos los elementos importantes que existan en una escena, ya sean personas, vehículos u otro tipo de objetos. El detector de objetos utilizado en el experimento original era el Faster R-CNN, sin embargo, decidimos sustituirlo por un modelo más actual y con otro tipo de características para ver cómo se comportaba. En la Figura 14 se indica que parte de la arquitectura ha sido modificada.

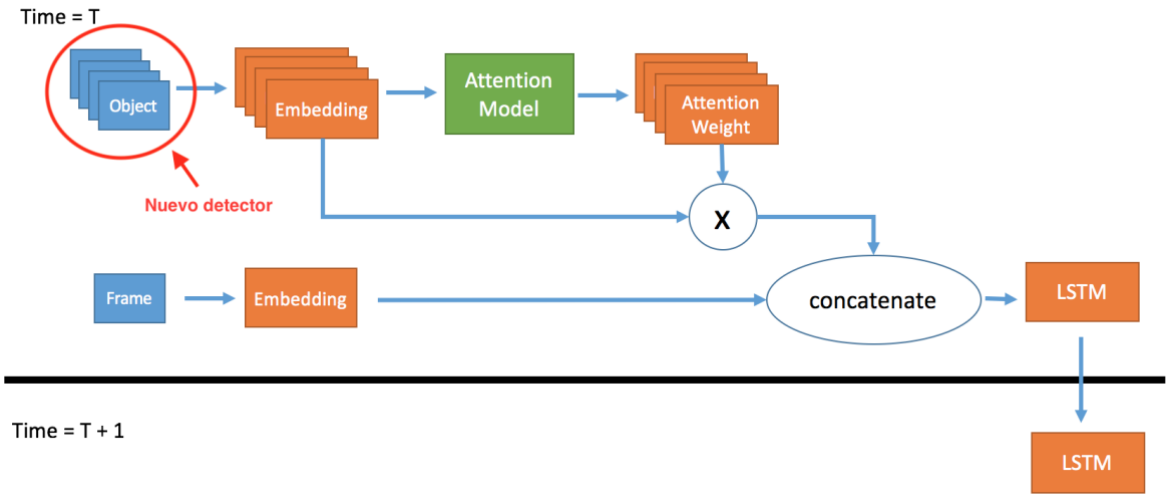

Figura 14: Diagrama de flujo del modelo con el nuevo detector

4.2.1 EfficientDet

Este nuevo detector fue propuesto en el *paper* "*EffiecientDet: Scalable and Efficient Object Detection*" [51] en el año 2020 por el equipo Google Brain. Según sus experimentos, el rendimiento de EfficientDet supera a modelos vanguardistas como YOLO o AmoebeaNet, y es por ello que decidimos que era el candidato perfecto para formar parte de nuestro trabajo [52]. Al igual que Faster R-CNN, configuramos EfficientDet para que detectara como máximo 19 objetos y se adaptará así a la arquitectura de nuestro código.

El modelo EfficientDet para devolver las coordenadas de "cajas" delimitadoras que recuadran las detecciones se basa principalmente en una red neuronal escalable conocida como EfficientNet y en la novedosa incorporación de una red de *features* bidireccionales (BiFPN) [53]. En la Figura 15 puede verse la arquitectura que acaba de ser mencionada.

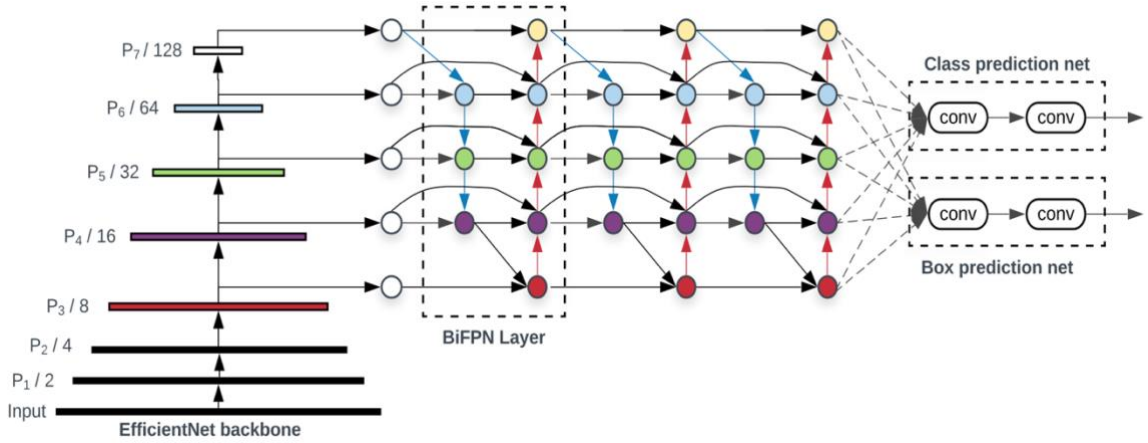

Figura 15: Arquitectura del detector EfficientDet. Procedente de [51]

4.3 Extracción de features

La extracción de *features* se trata de una operación que permite procesar la información procedente del detector de objetos (las coordenadas de las "cajas" que delimitan cada una de las detecciones) y obtener un vector de *features* (características) que define lo que aparece en cada "caja". El resultado de la red del algoritmo original era un vector de 4.096 *features* por cada detección, por lo que al procesar un *frame* de un vídeo se obtenía como máximo una matriz de tamaño 20 x 4.096, 19 vectores correspondientes a cada una de las detecciones y el restante correspondiente al procesamiento del *frame* global.

Para ampliar el espectro de los resultados de nuestro experimento decidimos reemplazar el módulo de extracción de *features* del algoritmo original por otro tipo de redes. En la Figura 16 se indica qué parte de la arquitectura ha sido modificada.

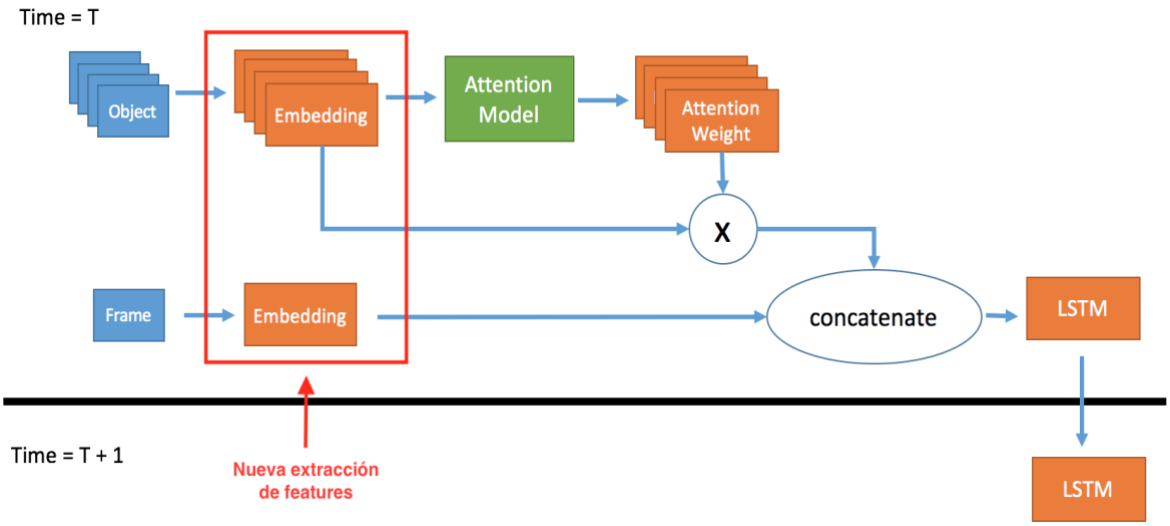

Figura 16: Diagrama de flujo del modelo con la nueva extracción de *features*

En los siguientes apartados serán explicadas las cuatro nuevas redes pre-entrenadas que seleccionamos para que formaran parte de nuestro experimento y que sustituirán al módulo de extracción de *features* original.

4.3.1 VGG16

VGG16 es una red neuronal convolucional desarrollada por el Visual Geometry Group de Oxford y presentada en el *paper* "*Very Deep convolutional networks for large-scale image recognition*" [43] en 2015. Se trata de una red muy popular ya que, aunque su estructura es bastante simple, es capaz de resolver prácticamente todos los problemas de clasificación. En el *dataset* ImageNet que contiene alrededor de 14 millones de imágenes correspondientes a 1000 clases alcanza una precisión de 92,7% [54].

VGG16 consta de 16 capas (13 convolucionales y 3 *fully-connected*) y como entrada recibe una imagen RGB de tamaño 224 x 224 x 3. La arquitectura que se acaba de comentar puede observarse en la Figura 17.

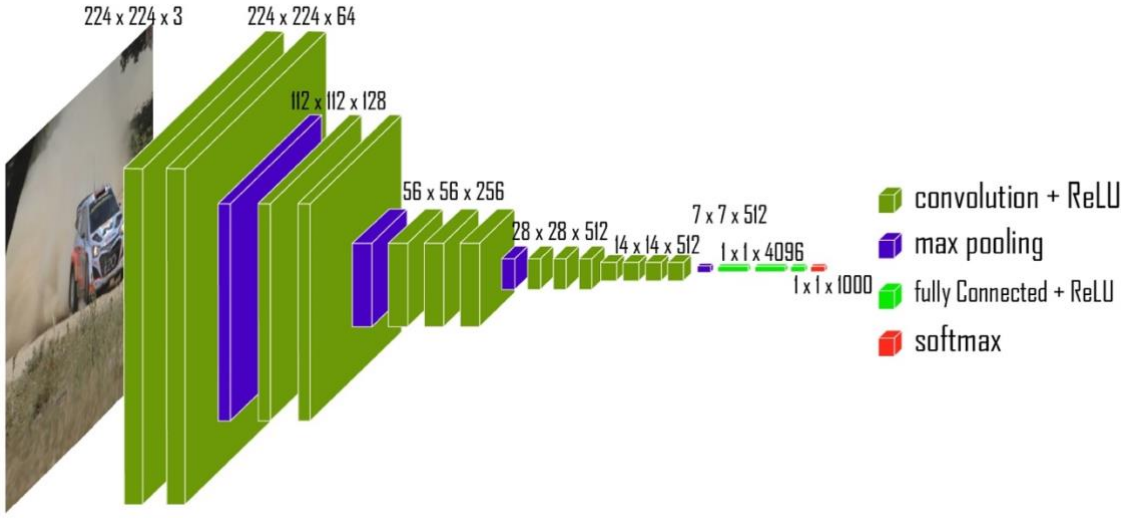

Figura 17: Arquitectura de la red VGG16. Procedente de [55]

En el algoritmo original se utiliza una red VGG, aunque sus desarrolladores no especifican qué tipo de configuración utilizan (VGG11, VGG13, VGG16 o VGG19). Por esta razón, decidimos volver a realizar el procesamiento con la VGG16 para obtener nuestros propios resultados con ella.

4.3.2 AlexNet

Se trata de una de una red anterior a VGG16, fue desarrollada por Alex Krizhevsky y sus compañeros y presentada en 2012 en el *paper* "*ImagenNet Classification with Deep Convolutional Neural Networks*" [56]. En ese momento, consiguió el mejor resultado al enfrentarse al *dataset* ImageNet, aunque años más tarde sería superado, aún así su rendimiento sigue siendo muy bueno..

AlexNet está formada por 5 capas convolucionales y 3 *fully-connected* y al igual que VGG16, a su entrada debe llegar una imagen RGB de tamaño 224 x 224 x 3. En la Figura 18 se presenta su arquitectura [57].

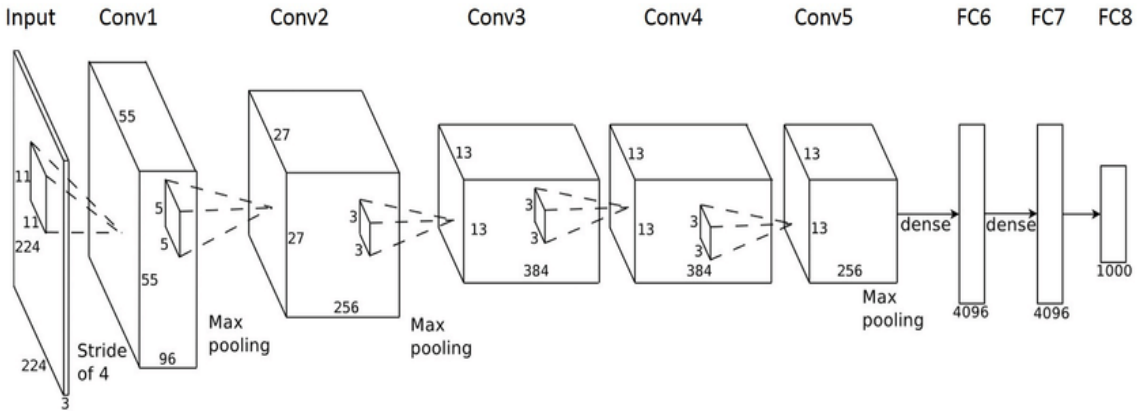

Figura 18: Arquitectura de la red AlexNet. Procedente de [58]

4.3.3 DenseNet

En el año 2017, el trabajo conjunto de la universidad de Cornell, la universidad de Tsinghua y Facebook AI Research culminó con la presentación en el *paper* "*Densely Connected Convolutional Networks*" [59] de un nuevo tipo de red llamada Dense Convolutional Network.

La principal característica de la DenseNet es que cada capa recibe información procedente de todas las capas anteriores y después de procesarla envía su salida a todas las capas posteriores, lo cual permite conseguir una alta eficiencia tanto computacional como en términos de memoria. En la Figura 19 se muestran las conexiones entre capas que existen habitualmente en una red DenseNet [60].

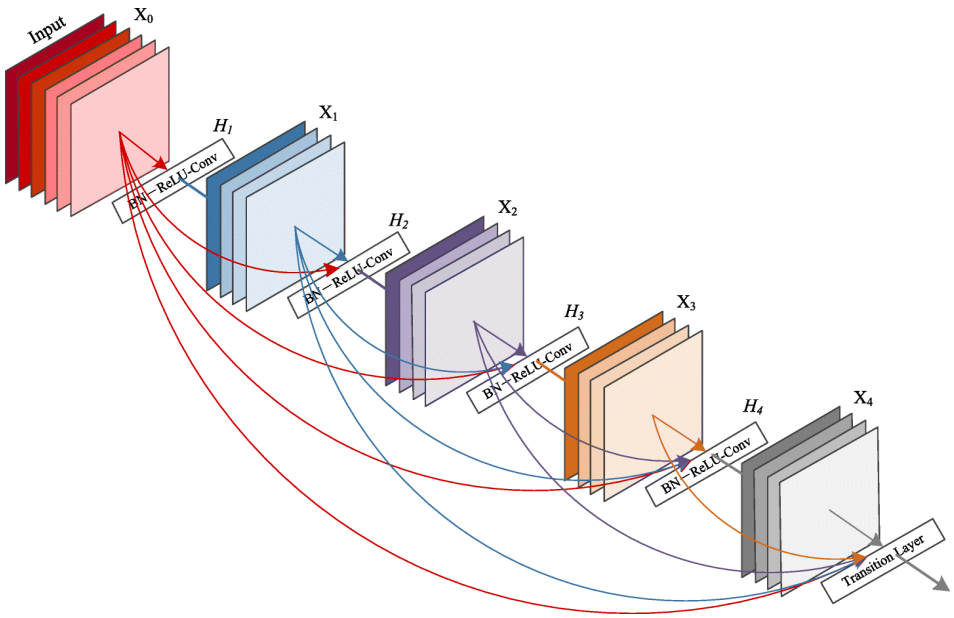

Figura 19: Conexiones entre capas de una red DenseNet. Procedente de [61]

Existen diferentes versiones de redes DenseNet dependiendo del número de capas neuronales que la formen. En nuestro caso decidimos utilizar DenseNet-161, que como su nombre indica tiene 161 capas.

4.3.4 ResNeXt

Esta última red es junto con DenseNet la más moderna que hemos utilizado. Fue desarrollada por la Universidad de California San Diego junto con Facebook AI Research y presentada en el *paper* "*Transformations for Deep Neural Networks*" [62] en 2017.

Se basa en la red ResNet, pero con una evolución, incorpora una nueva dimensión denominada *Cardinality* que permite controlar el número de transformaciones complejas. Esta modificación, en teoría, consigue mejorar el rendimiento de ResNet [63].

Al igual que para DenseNet, para ResNeXt existen dos versiones dependiendo del número de capas de la red. Decidimos utilizar la versión con 50 capas que se denomina resnext50_32x4d.

5 Integración, pruebas y resultados

5.1 Introducción

A lo largo de este capítulo presentaremos los resultados obtenidos con las diferentes redes y el nuevo detector y los compararemos con los resultados originales del *paper* [6]. Sin embargo, antes de abordar este tema, contextualizaremos los experimentos refiriéndonos a la naturaleza de los *datasets* utilizados y los cálculos realizados para obtener estos resultados.

5.2 Dataset original

Para el experimento original, los desarrolladores querían utilizar un *dataset* que fuese muy desafiante y que por tanto demostrase la calidad del algoritmo. Para ello, reunieron 620 vídeos de *dashcam* (videos grabados con una cámara colocada en el salpicadero del vehículo) de alta calidad que incluían accidentes en el entorno del vehículo y que habían sido filmados a 20 *frames* por segundo en varias de las ciudades más importantes de Taiwán. En estos 620 vídeos marcaron manualmente lo ubicación temporal de los accidentes y las coordenadas de las "cajas" que recuadran cada uno de los objetos de un *frame* para poder entrenar de forma supervisada el modelo y luego evaluarlo.

Lo que hace que estos vídeos sean tan desafiantes es que en Taiwán las señales de tráfico son bastante complicadas comparadas con las europeas, además las calles de las ciudades suelen estar abarrotadas de personas y vehículos por lo que existe mucho "movimiento" y por ende, la probabilidad de accidente es muy alta.

A partir de los 620 vídeos extrajeron 1750 *clips* de 5 segundos (100 *frames*) cada uno. 620 de estos *clips* contienen un accidente alrededor del vehículo en sus 10 últimos *frames* (los llamaron positivos) y 1130 no contienen ningún accidente (los llamaron negativos). Por último, dividieron de forma aleatoria el *dataset* entre *training* y *testing*, eso sí, siendo el número de *clips* de *training* alrededor del triple del de *testing*. La estructura del *dataset* original se puede observar en la Tabla 1.

	TRAINING	TESTING	TOTAL
POSITIVO	455	165	620
NEGATIVO	829	301	1130
TOTAL	1284	466	1750

Tabla 1: Estructura del *dataset* original. Se indica el número de clips, entre positivos (con accidente) y negativos (sin accidente) para *training* y *testing*

5.3 Subset escogido

Cuando comenzamos a realizar nuestro propio experimento modificando los módulos del detector de objetos y de la extracción de *features* del algoritmo, nos dimos cuenta de que, debido a los limitados recursos técnicos de los que disponíamos, era una utopía trabajar con el *dataset* original. Junto a los casi 2.000 *clips* (unos 3 GB), para ejecutar el algoritmo, también sería necesario almacenar por cada *clip*, las coordenadas de las detecciones, sus correspondientes *features* y la etiqueta positiva o negativa. Esta información ocuparía alrededor de 115 GB, por lo que, si lo multiplicáramos por 4, debido a las 4 nuevas redes, necesitaríamos unos 500 GB. Además, como utilizamos un nuevo detector, habría que volver a obtener toda esta información, así que el espacio necesario se duplicaría. Por último, para procesar todos los *clips* y obtener las *features*, sólo con una red, necesitaríamos alrededor de 4 meses debido al coste computacional de las operaciones.

Como es entendible, optamos por reducir el *dataset* para que se adaptara a nuestras limitaciones. Para ello, decidimos trabajar únicamente con los 466 *clips* que en principio eran de *testing*. Los dividimos aproximadamente en 1/3 para *testing* y 2/3 para *training*. Aunque el procesamiento aún duró más de un mes y tuvimos algunos problemas de almacenamiento, finalmente conseguimos realizar nuestros propios experimentos. La estructura del *subset* escogido se puede observar en la Tabla 2.

	TRAINING	TESTING	TOTAL
NUEVO SUBSET	300 (2/3 *testing* original)	166 (1/3 *testing* original)	466 (*testing* original)

Tabla 2: Estructura del *subset* escogido. Se indica en número de *clips* como se distribuye el *dataset* original de *testing* en los nuevos *subsets* de *training* y *testing*

5.4 Métrica

Durante el procesamiento de cada uno de los *clips* a los que nos hemos referido anteriormente, por cada *frame*, se calcula la probabilidad de posible accidente en torno al vehículo, si esta probabilidad es mayor o igual que un umbral, el algoritmo señalará que un accidente ocurrirá y si la probabilidad es menor que el umbral, el algoritmo señalará que por el momento no se prevé ningún accidente. Dependiendo de estas predicciones y de las anotaciones originales de los *clips*, existen 4 casos:

- **True Positive (TP):** El algoritmo notifica un futuro accidente en el *clip* y además este *clip* ha sido marcado como positivo (realmente ocurre un accidente, ver Sección 5.2).

- **False Positive (FP):** El algoritmo notifica un futuro accidente en el *clip*, sin embargo, este *clip* ha sido marcado como negativo (no ocurre un accidente, ver Sección 5.2).

- **True Negative (TN):** El algoritmo no notifica un futuro accidente en el *clip* y además este *clip* ha sido marcado como negativo (no ocurre un accidente, ver Sección 5.2).

- **False Negative (FN):** El algoritmo no notifica un futuro accidente en el *clip*, sin embargo, este *clip* ha sido marcado como positivo (realmente ocurre un accidente, ver Sección 5.2).

A partir de estos cuatro casos, las dos variables más habituales para medir el rendimiento de un algoritmo son las siguientes:

- **Precision (Precisión):** Indica qué proporción de identificaciones positivas fueron correctas. Su fórmula es $\frac{TP}{TP+FP}$.

- **Recall (Exhaustividad):** Indica la relación que existe entre las identificaciones positivas y todos los casos positivos reales. Su fórmula es $\frac{TP}{TP+FN}$.

5.4.1 Area Under the Curve (AUC)

El principal problema que presentan los parámetros de Precisión y Recall es que varían profundamente según el umbral que se decida elegir. Con el objetivo de representar todo el espectro de nuestro experimento, el conjunto de umbrales elegidos lo forman la probabilidad de accidente de cada uno de los *frames* de todos los *clips* de *testing*. Para cada umbral se obtiene un valor de Precisión y Recall teniendo en cuenta todo el conjunto de *testing*.

Utilizamos todos estos datos para generar un resultado denominado Area Under the Curve que, mediante el cálculo del área debajo de la curva Precisión vs Recall, permite determinar la capacidad de un clasificador para distinguir entre clases.

5.4.2 Tiempo Medio al Accidente

Cuando el algoritmo notifica un futuro accidente en el instante x y el *clip* contiene realmente un accidente en el instante t, nuestro modelo genera el parámetro Tiempo al Accidente, el cual nos permite conocer el tiempo que existe desde que se prevé por primera vez el accidente hasta que sucede. La fórmula para el Tiempo al Accidente es $t - x$. Al igual que para Precisión y Recall, generamos un valor de Tiempo al Accidente para cada umbral.

En los resultados de la siguiente sección presentamos la media de todos los Tiempo al Accidente y el Tiempo al Accidente al 80% de Recall como ejemplo concreto, ya que consideramos un rendimiento razonable de un algoritmo cuando su Recall se encuentra en torno al 80%.

5.5 Resultados

En el capítulo de Desarrollo se han detallado minuciosamente los cambios que hicimos en el algoritmo original para realizar nuestro propio experimento. A continuación, se exponen todos los resultados obtenidos a partir de estas modificaciones y son comparados con los logros del modelo original.

Junto con el *dataset* que hemos comentado al inicio de este capítulo, los desarrolladores del algoritmo original ofrecen la posibilidad de descargar toda la información necesaria (*features*, etiquetas, coordenadas de detecciones) para poder ejecutar su modelo tal y como ellos lo diseñaron. Como punto de partida a nuestro experimento decidimos obtener sus resultados y se muestran en la Tabla 3:

	AUC	TIEMPO MEDIO AL ACCIDENTE (seg)	TIEMPO AL ACCIDENTE AL 80% DE RECALL (seg)
Experimento original	0.7258	1.153	1.715

Tabla 3: Resultados del experimento original

Ya ha sido mencionado que debido a nuestras limitaciones técnicas era imposible utilizar todo el *dataset* disponible por lo que, para comparar nuestro estudio de igual a igual con el realizado inicialmente, decidimos, de manera similar, probar el algoritmo original con la información descargada y procedente del experimento realizado mediante el detector Faster R-CNN y una red VGG (no ha sido especificado por los desarrolladores cuál), pero sólo con el *subset* que nosotros vamos a utilizar. Estos resultados, a partir de este punto, se denominarán: Original nuevo Subset.

Además de los resultados de Original nuevo Subset, también presentamos los de Vgg16, AlexNet, DenseNet, ResNeXt, que se corresponden con los resultados obtenidos al extraer las *features* con cada una de esas redes. Por último, estos 5 experimentos se realizaron tanto para el detector Faster R-CNN (el original) como para el EfficentDet (el nuevo).

5.5.1 Resultados Faster R-CNN

Los resultados obtenidos con el detector Faster R-CNN para cada una de las diferentes redes quedan registrados en la Tabla 4:

Faster R-CNN	Original nuevo Subset	Vgg16	AlexNet	DenseNet	ResNeXt
AUC	0.6384	0.8628	0.8777	0.3810	0.3648
TIEMPO MEDIO AL ACCIDENTE	1.743	1.901	3.687	0.2485	0.2441
TIEMPO AL ACCIDENTE AL 80% DE RECALL	3.005	4.734	4.491	0.3002	0.2612

Tabla 4: Resultados con el detector Faster R-CNN

Para completar el informe sobre los resultados con este detector, se incluyen las gráficas de Precisión vs Recall (Figura 20) y de Tiempo Medio al Accidente vs Recall (Figura 21) que permiten comparar de forma más gráfica el rendimiento de cada experimento.

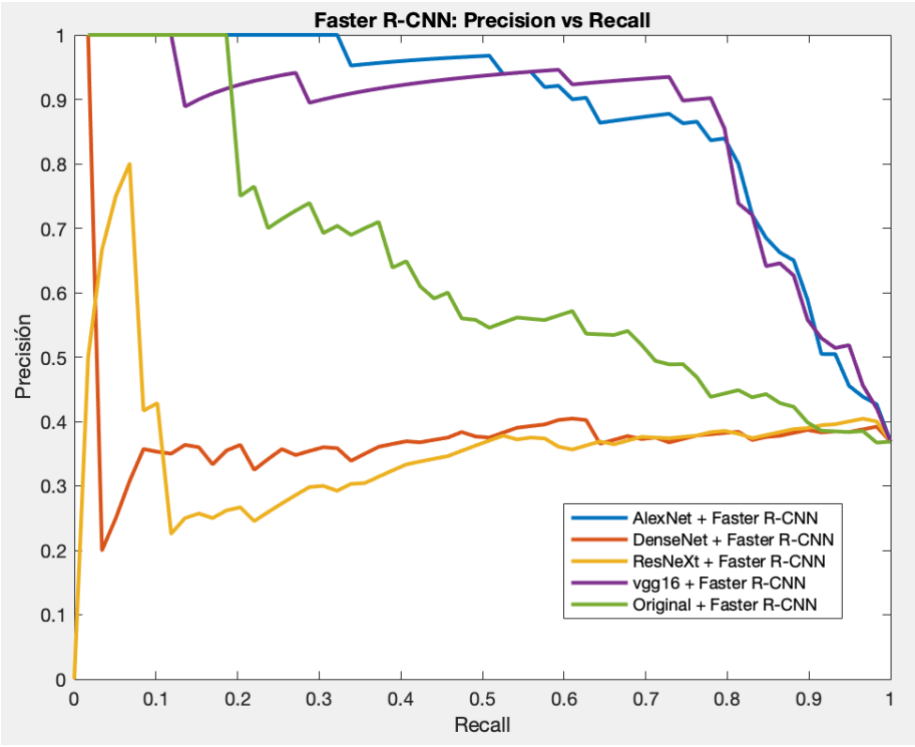

Figura 20: Gráfica Precisión vs Recall para el detector Faster R-CNN

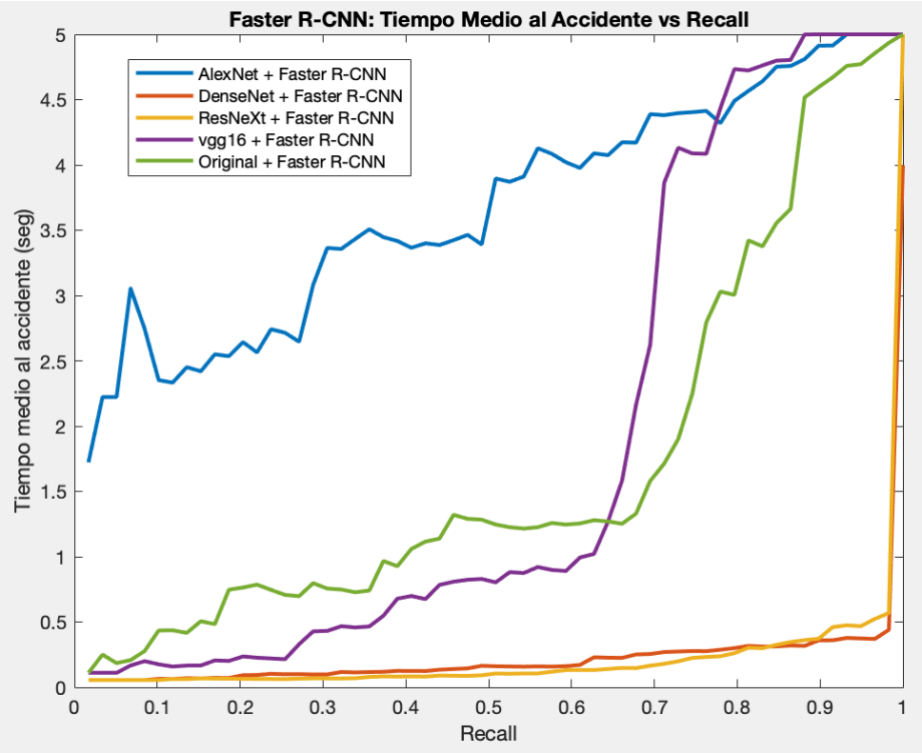

Figura 21: Gráfica Tiempo Medio al Accidente vs Recall para el detector Faster R-CNN

5.5.2 Resultados EffientDet

Los resultados obtenidos con el detector EffientDet para cada una de las diferentes redes quedan registrados en la Tabla 5. Es importante destacar que en este caso no se presentan los datos Original nuevo Subset porque esos resultado se obtienen con información había sido descargada y no era posible modificarla para cambiar la parte de las detecciones:

EfficentDet	Vgg16	AlexNet	DenseNet	ResNeXt
AUC	0.4917	0.5309	0.3688	0.3912
TIEMPO MEDIO AL ACCIDENTE	2.9667	2.813	0.3116	0.2724
TIEMPO AL ACCIDENTE AL 80% DE RECALL	4.652	4.734	0.3144	0.1135

Tabla 5: Resultados para el detector EfficientDet

Al igual que para Faster R-CNN, también se incluyen las gráficas de Precisión vs Recall (Figura 22) y de Tiempo Medio al Accidente vs Recall (Figura 23).

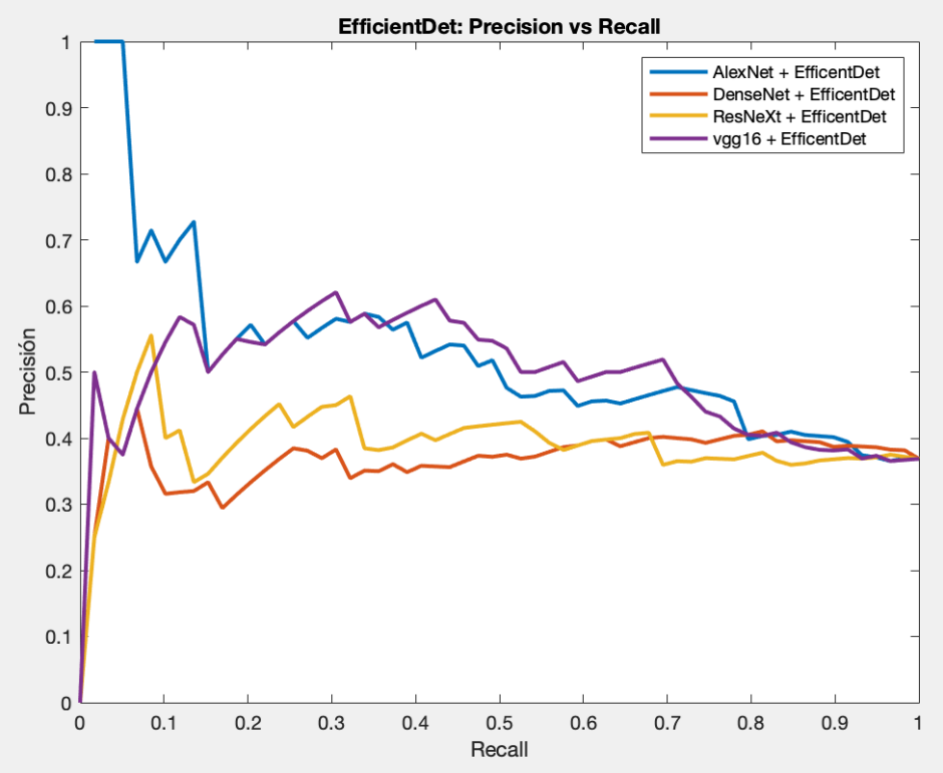

Figura 22: Gráfica Precisión vs Recall para el detector EfficientDet

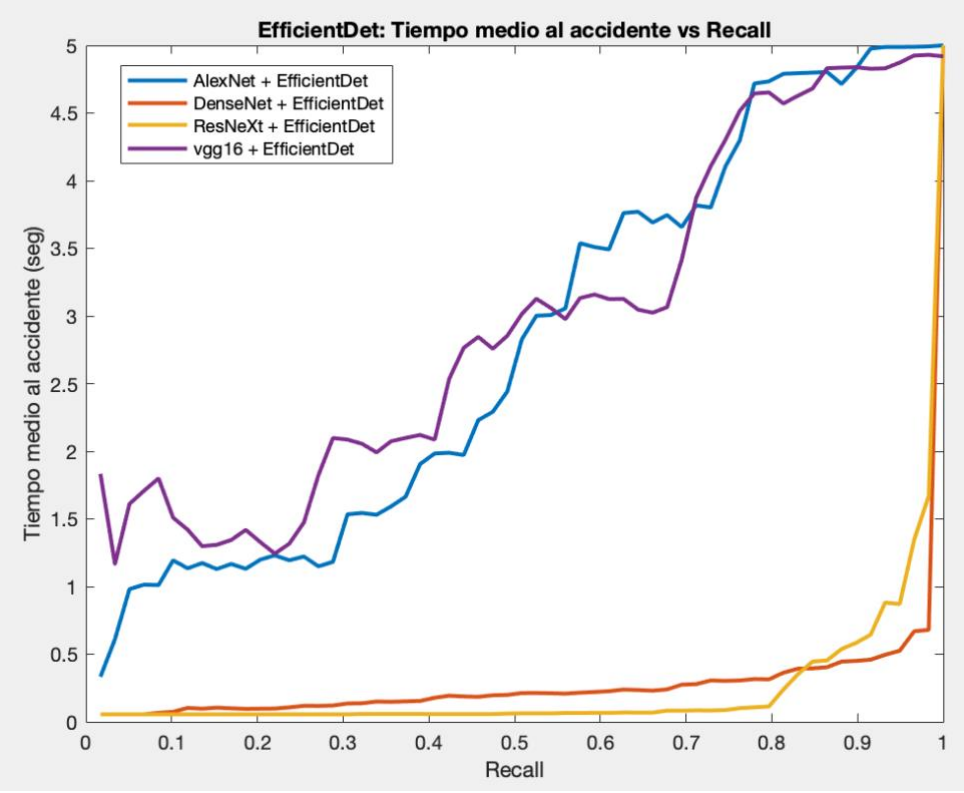

Figura 23: Gráfica Tiempo Medio al Accidente vs Recall para el detector EfficientDet

5.5.3 Interpretación de los resultados

Al analizar de forma global todos los resultados, la primera impresión que se desprende es que para el experimento concreto en el que hemos estado trabajando, el rendimiento en términos generales del detector Faster R-CNN supera con claridad al detector EfficientDet. En concreto, respecto a Area Under the Curve, las diferencias son abismales, siendo el mejor resultado para EfficientDet (≈0.53) alrededor de un 40% peor que el mejor resultado para EffiecientDet (≈0.88).

El siguiente punto a destacar es el diferente comportamiento de cada una de las redes para extraer las *features* de las detecciones. Sin duda, se observan dos grupos bien diferenciados, las redes Vgg16 y AlexNet presentan un rendimiento más que aceptable (sobre todo para Faster R-CNN) y muy similares en todos los parámetros de evaluación, sin embargo, las redes DenseNet y ResNeXt ni siquiera son capaces de predecir un accidente con medio segundo de antelación ni alcanzan un 0.5 en Area Under the Curve, por lo que su uso en esta tarea concreta no parece muy recomendable. Estos malos resultados pueden deberse a que utilizamos la misma configuración de entrenamiento para el modelo (misma tasa de entrenamiento y número de épocas) con todas las redes y de esta forma, realizar todos los experimentos en las mismas condiciones. Es probable que las redes DenseNet y ResNeXt no se hayan adaptado tan bien como Vgg16 y AlexNet a esta configuración y de ahí su pobre rendimiento.

Otro de los aspectos importantes de este experimento es la capacidad de predecir con antelación y de forma acertada un accidente alrededor del vehículo. Aunque el mejor resultado para el Tiempo Medio al Accidente se consigue con Faster R-CNN (≈3.7 seg), la situación es mucho más ajustada entre los dos detectores, ya que los dos siguientes resultados más relevantes se obtienen gracias a EfficientDet. Casualmente, y corroborando esta igualdad en términos de anticipación temporal, en el caso concreto fijando un Recall de 80%, el mejor resultado es el mismo para ambos detectores, pero con diferentes redes (≈4.7 seg).

Para finalizar, vamos a comparar los resultados obtenidos con la información que nosotros generamos y los obtenidos con la información proporcionada por los desarrolladores del algoritmo original. Respecto a Area Under the Curve nuestros modelos con el detector Faster R-CNN y las redes Vgg16 y AlexNet superan ampliamente a los datos obtenidos por el modelo original tanto para el *dataset* completo como para el *subset* reducido. En cuanto al Tiempo al Accidente, nuestros resultados para ambos detectores y las redes Vgg16 y AlexNet son bastante similares y de nuevo mejoran claramente a los resultados con la arquitectura original.

6 Conclusiones y trabajo futuro

6.1 Conclusiones

Esencialmente el objetivo de este trabajo era introducir al lector a una tecnología tan vanguardista como es la detección de accidentes de procesado de vídeo y realizar un análisis detallado y minucioso de todo lo que la rodea, para luego centrarnos en un caso muy concreto, pero con multitud de posibilidades y realizar un experimento que nos permitiera ser conscientes del tremendo potencial que posee.

A lo largo de todo el documento, han sido explicadas las situaciones en las que podría aplicarse esta tecnología, los principales conceptos matemáticos en los que se basa o los algoritmos más populares que se utilizan en este tipo de tareas, hasta finalmente presentar nuestros experimentos y comparar sus resultados con otros que ya habían sido obtenidos.

Una vez realizada la tarea de contextualizar la tecnología durante los dos primeros capítulos del trabajo, en la segunda parte nos centramos en el desarrollo y el diseño de un modelo donde poder aplicar lo aprendido al estudiar el estado del arte. Basándonos en el algoritmo propuesto en el *paper* "*Anticipating Accidents in Dashcam Videos"* [6] con Faster R-CNN y VGG, desarrollamos nuestro propio modelo para detectar accidentes que ocurren alrededor de un vehículo para poder anticiparse a ellos. Modificamos los módulos de detección de objetos con un nuevo detector (EfficientDet) y el de extracción de *features* con cuatro redes diferentes (AlexNet, VGG16, DenseNet y ResNeXt).

Tras el análisis de los resultados se llegó a la conclusión que las combinaciones de módulos que maximizan el Area Under the Curve la forman Faster R-CNN + VGG16 y Faster R-CNN + AlexNet. Por otra parte, si se está más interesado en anticipar con margen un posible accidente, la utilización del detector Faster R-CNN como el de EfficienDet (siempre con las redes VGG16 y AlexNet) son igual de válidas ya que presentan resultados similares. Por último, nuestro experimento también puso de manifiesto que las redes DenseNet y ResNeXt no deben emplearse (con esta configuración de entrenamiento) para esta tarea concreta ya que sus resultados en todos de los parámetros de estudio son muy mejorables.

6.2 Trabajo futuro

Una de las principales razones por las que escogimos basarnos en el modelo propuesto en el *paper* "*Anticipating Accidents in Dashcam Videos"* [6] es que era muy fácil separar su estructura por módulos y poder así modificar las etapas de detección de objetos y de extracción de *features*. Como es entendible, esta característica ofrece multitud de posibilidades para trabajar en el futuro con él.

Siguiendo una línea continuista con nuestro trabajo, se podría ampliar el espectro de resultados realizando experimentos con nuevos detectores (MobileNet, YOLO o CenterNet) y con nuevas redes (ResNet, Inception v3 o GoogLeNet) para luego compararlos con los presentados en el capítulo anterior. También sería valioso conocer cómo se comportan las redes al utilizar distintos parámetros de entrenamiento (tasa de entrenamiento y número de épocas).

Asimismo, ya se ha comentado que debido a nuestras limitaciones técnicas no pudimos utilizar todo el *dataset* del que disponíamos. Por ello, los resultados serían mucho más precisos y reveladores si el modelo pudiera ser entrenado y testeado con el *dataset* completo. Por otra parte, aunque el conjunto de *clips* con el que trabajamos ya es muy desafiante, otra posibilidad de desarrollo sería analizar el comportamiento del algoritmo con un nuevo *dataset* con características similares pero que hubieran sido filmados en otro tipo de situaciones.

Si se buscan tareas más complejas a las que enfrentarse utilizando como base este trabajo, sería interesante no sólo considerar el entorno de un vehículo como potencial generador de accidentes sino también el propio vehículo, para ello se necesitaría un nuevo *dataset* en el que ocurrieran este tipo de situaciones y modificar el algoritmo para el cálculo de la probabilidad de accidente. Por último, y respecto al cálculo de la probabilidad de accidente, podría incluirse información de la vía por la que circula el vehículo (número de carriles, cantidad de vehículos o el estado de conservación) como una variable más a tener en cuenta en un posible accidente.

Referencias

[1] AutoBild. Los mejores asistentes a la conducción. (2019). https://www.autobild.es/practicos/mejores-asistentes-conduccion-539859

[2] Balbás, A., Domínguez, M. & Espinosa, M. El volante en la conducción autónoma. Revista Iberoamericana de Ingeniería Mecánica, Vol. 23, Nº 2, pp. 92-93. (2019).

[3] Km 77. Conducción autónoma. (2021). https://www.km77.com/reportajes/varios/conduccion-autonoma-niveles

[4] DGT. Presentación de las cifras de siniestralidad vial de 2019. Anexo estadístico Agosto 2020. (2020).

[5] epdata. Accidentes de tráfico en datos y estadísticas. (2021). https://www.epdata.es/datos/accidentes-trafico-datos-estadisticas/65/espana/106

[6] Chan, F., Chen, Y., Xiang, Y. & Sun, M. Anticipating Accidents in Dashcam Videos. Asian Conference on Computer Vision. (2016).

[7] Pinar, A., Cicekli, I., Akman, V. Turing Test: 50 years later. Minds & Machines 10. (2000).

[8] Pascual, J. Computer Hoy. Inteligencia artificial; qué es, cómo funciona y para que se está utilizando. (2019). https://computerhoy.com/reportajes/tecnologia/inteligencia-artificial-469917

[9] Barceló, M. Inteligencia Artificial. UOC Papers. (2002).

[10] Bagnato, J. Aprende Machine Learning en Español: Teoría + Práctica Python. (2020).

[11] Mitchell, T. Machine Learning. (1997).

[12] Alpaydin, E. Machine Learning: The new AI. (2016).

[13] Simeone, O. A Very Brief Introduction to Machine Learning With Applications to Communication Systems. IEEE Transactions on Cognitive Communications and Networking, Vol. 4, No. 4, pp. 648-664. (2018).

[14] Mueller, J. & Massaron, L. Deep Learning for Dummies. (2019).

[15] LeCun, Y., Bengio, Y. & Hinton, G. Deep learning. Nature, Vol. 521, pp. 436–444. (2015).

[16] Yoshida, S. R. Computer Vision. Computer Science, Technology and Applications. (2011)

[17] Puentes, N. Qué es Computer Vision. (2021). https://www.nebulova.es/blog/computer-vision

[18] Ionescu, R., Popescu, M. Knowledge Transfer between Computer Vision and Text Mining. (2016).

[19] Dreyfus, G. Neural Networks: An Overview. (2005).

[20] Gurney, K. An Introduction to Neural Networks. (1997).

[21] Larrañaga, P., Inza, I. & Moujahid, A. Redes Neuronales. Departamento de Ciencias de la Computación e Inteligencia Artificial. Universidad del País Vasco.

[22] Salas, R. Redes Neuronales Artificiales. Departamento de Computación. Universidad de Valparaíso. (2004)

[23] Florez, R. & Fernández, J. M. Las Redes Neuronales Artificiales. Fundamentos tecnológicos y aplicaciones prácticas. (2008).

[24] San Miguel, J.C. Diapositivas Reconocimiento y detección en imágenes basado en redes neuronales (Deep Learning) Fundamentos. Tratamiento de Señales Visuales. (2019).

[25] Du, K.L. & Swamy, M.N.S. Multilayer Perceptrons. Architecture and Error Backpropagation. (2019).
[26] Mohamed, H., Negm, A., Zahran., M. & Saavedra, O. Assessment of artificial neural network for bathymetry estimation using high resolution satellite imagery in shallow lakes: case study El Burullus Lake. (2015).
[27] Olah, C. Neural Networks, Manifolds, and Topology. (2014).
[28] Llano, L., Hoyos, A., Arias, F. & Velásquez, J. Comparación del desempeño de funciones de activación en redes feedforward para aproximar funciones de datos con y sin ruido. Avances en Sistemas e Informática. Vol 4 (2). (2007).
[29] Gavilán, I. Catálogo de componentes de redes neuronales (II): funciones de activación. (2020). https://ignaciogavilan.com/catalogo-de-componentes-de-redes-neuronales-ii-funciones-de-activacion/
[30] Chauvin, Y. & Rumelhart, D. Backpropagation. Theory, Architectures and Applications. (1995).
[31] Rumelhart, D., Hinton, G. & Williams, R. Learning representations by back-propagating errors. Nature. Vol. 323, pp. 533–536. (1986)
[32] Sewak, M., Karim, R. & Pradeep, P. Practical Convolutional Neural Networks: Implement Advanced Deep Learning Models Using Python. (2018).
[33] Michelucchi, U. Advanced Applied Deep Learning: Convolutional Neural Networks and Object Detection. (2019).
[34] Khan, S., Rahmani, H., Afaq, S. & Bennamoun, H. A Guide to Convolutional Networks for Computer Vision. (2018).
[35] Aprende Machine Learning. ¿Cómo funcionan las Convolutional Neural Networks? Visión por Ordenador. (2018). https://www.aprendemachinelearning.com/como-funcionan-las-convolutional-neural-networks-vision-por-ordenador/
[36] Vasilev, I. A Deep Learning Tutorial: From Perceptrons to Deep Networks. https://www.toptal.com/machine-learning/an-introduction-to-deep-learning-from-perceptrons-to-deep-networks
[37] San Miguel, J.C. Diapositivas Redes Neuronales Convolucionales parte 2. Tratamiento de Señales Visuales. (2019).
[38] Pang, Y. & Cao, J. Deep Learning in Object Detection. Deep Learning in Object Detection and Recognition. pp. 19-57. (2019).
[39] Girshick, R. Faster R-CNN. (2015).
[40] Redmon, J. & Farhadi, A. YOLO9000: Better, Faster, Stronger. (2016).
[41] Deepak, D. An evaluation of deep learning based object detection strategies for threat object detection in baggage security imagery. Pattern Recognition Letters. Vol. 120, pp. 112-119. (2019).
[42] Ren, S., He, K., Girshick, R. & Sun, J. Faster r-cnn: Towards real-time object detection with region proposal networks. (2015).
[43] Simonyan, K. & Zisserman, A. Very deep convolutional networks for large-scale image recognition. (2015).
[44] Smallcorgi. https://github.com/smallcorgi/Anticipating-Accidents#readme. ()
[45] Xu, K., Ba, J., Kiros, R., Courville, A., Salakhutdinov, R., Zemel, R. & Bengio, Y. Show, attend and tell: Neural image caption generation with visual attention. (2015).
[46] Géron, A. Neural networks and deep learning. (2018).
[47] Bonet, I., Salazar, S., Rodríguez, A. & Grau, R. Redes neuronales recurrentes para el análisis de secuencias. Revista Cubana de Ciencias Informáticas. Vol. 1, Nº 4, pp. 48-57. (2007).

[48] Church, C. Implementation of RNN, LSTM, and GRU. https://towardsdatascience.com/implementation-of-rnn-lstm-and-gru-a4250bf6c090. (2019).
[49] Pascanu, R., Mikolov, T. & Bengio, Y. On the difficulty of training Recurrent Neural Networks. (2013).
[50] Mañas, A. LSTM univariado. Notas sobre pronóstico del flujo de tráfico en la ciudad de Madrid. (2019).
[51] Tan, M., Pang, R. & Le, Q. EfficientDet: Scalable and Efficient Object Detection. (2020).
[52] EfficientDet: Guide to State of The Art Object Detection Model. https://analyticsindiamag.com/efficientdet/. (2020)
[53] Tan, M. & Y, A. EfficientDet: Towards Scalable and Efficient Object Detection. https://ai.googleblog.com/2020/04/efficientdet-towards-scalable-and.html. (2020).
[54] Thakur, R. Step by step VGG16 implementation in Keras for beginners. https://towardsdatascience.com/step-by-step-vgg16-implementation-in-keras-for-beginners-a833c686ae6c. (2019).
[55] VGG-16 | CNN model. https://www.geeksforgeeks.org/vgg-16-cnn-model/. (2020).
[56] Krizhevsky, A., Sutskever, I. & Hinton, G. ImageNet Classification with Deep Convolutional Neural Networks. (2012).
[57] Alom, M., Taha, T., Yakopcic, C., Westberg, S., Sidike, P., Nasrin, M., Van Essen, B., Awwal, A. & Asari, V. The History Began from AlexNet: A Comprehensive Survey on Deep Learning Approaches. (2018).
[58] Shehata, S. Using mid- and high-level visual features for surgical workflow detection in cholecystectomy procedures. https://www.researchgate.net/figure/An-illustration-of-the-architecture-of-AlexNet-deep-convolutional-neural-network_fig3_308880040. (2016)
[59] Huang, G., Liu, Z. & v. d. Marteen, L. Densely Connected Convolutional Networks. (2017).
[60] Tsang, S. Review: DenseNet — Dense Convolutional Network (Image Classification). https://towardsdatascience.com/review-densenet-image-classification-b6631a8ef803. (2018).
[61] AI Pool. https://ai-pool.com/m/densenet-1568742493. (2019).
[62] Xie, S., Girshick, R., Dollár., Tu, Z. & He, K. Aggregated Residual Transformations for Deep Neural Networks. (2017).
[63] Tsang, S. Review: ResNeXt — 1st Runner Up in ILSVRC 2016 (Image Classification). https://towardsdatascience.com/review-resnext-1st-runner-up-of-ilsvrc-2016-image-classification-15d7f17b42ac. (2018).

Glosario

TFG	Trabajo Fin de Grado
IA	Inteligencia Artificial
DSA-RNN	Dynamic-Spatial-Attention Recurrent Neural Network
LSTM	Long Short-Term Memory
TP	True Positive
FP	False Positive
TN	True Negative
FN	False Negative
AUC	Area Under the Curve